ESCROW: WHEN A RECORD STREAM HAS EARNED A NEW NODE

Sourav Banerjee
Indian Institute of Technology Kharagpur & Shunya Labs

ABSTRACT

Each arrival in a stream of records raises the same question: does the record join a node that already exists, or has the stream earned a new one? The rules in use answer with a constant chosen by hand: a distance cut-off in streaming clustering, the penalty the small-variance limit leaves in DP-means and BP-means, or the sampling temperature of a language-model pipeline. In the second case the constant is what a deletion leaves behind. Before the limit is taken the new-node penalty is a codelength, the limit erases the part of it that could have been computed, and we do not take the limit. We read the price of a node off a code that predicts each record before seeing it, so the price is exact and grows with the logarithm of the stream, and no quantity in the rule is calibrated on data. No fresh node can pay that price on its first record under any valid code, so evidence accrues in escrow against a node that does not yet exist and the node is created when its account covers its price. On pure noise it creates nothing, where streaming decision trees grow to a mean of ninety-one false nodes, and it recovers planted structure exactly in all twenty arrival orders. On raw Wikipedia infoboxes its agreement with the category labels runs from 0.64 to 0.99 with the record order, which is the measured cost of a one-pass greedy sequence, and a language model given the same records is more accurate there but returns a different graph on every seed. Two free-text benchmarks are negatives and reported as such: on one we sit at the benchmark's no-merge floor together with the oracle-tuned embedding baselines, at macro F1 0.016; on the other, where identity is a title one token apart, we score 0.001 against their 0.78.

1 INTRODUCTION

A stream of records arrives and does not stop. A record might be a product listing, a page with an infobox, a song, or a person, and whatever it is, it arrives as a small set of named parts, one key and one value at a time. We want a graph of what the stream contains, built while the stream is still running, with a node for each kind of thing in it and an edge from every record to the nodes it belongs to. Building such a graph means answering the same question at every arrival. Does this record belong to a node that already exists, or has the stream now earned a new node?

The word worth defending there is earned. If a node exists because the data paid for it then there is a price, and something has to say what that price is. Three families of rules decide node existence today, and not one of them computes it. A streaming clustering rule opens a node when the distance from a record to the nearest one exceeds a chosen value; the rule is deterministic and easy to audit, and the value is a choice. The same shape turns up far from clustering, since XGBoost adds a leaf only when the gain exceeds a per-leaf charge γ whose default is zero and for which its own paper reports no value (Chen & Guestrin, 2016). DP-means and BP-means come closer, because they begin from a model of how many groups a stream contains (Kulis & Jordan, 2012; Broderick et al., 2013b), but they obtain their new-node penalty λ by sending the noise that model assumes to zero. Language-model pipelines (Bai et al., 2025; Hongwimol et al., 2026) state no price at all. They ask

Code, experiments and this paper: `https://github.com/Shunyalabsai/ESCROW`

a model, record by record, and keep what it samples, so two runs over the same data can return different graphs (Figure 1).

The Bayesian rules are the interesting failure, because their constant was not there at the start. Before the limit is taken, the new-node penalty is a codelength, a quantity the model works out for itself from what it has already seen. The limit multiplies the objective by the assumed noise variance, and Lemma C.1 shows what that does: it annihilates every part of the penalty that could have been computed and keeps only the part a person inserted. So λ is not a computed quantity that happens to be hard to compute. It is what a deletion left behind.

So we do not take the limit, and that leaves us a way to compute the price. A probability model of a stream is also a code for it, because a prediction that gives probability P to what happens can write it down in $-\log_2 P$ bits, and once everything is in bits the accounting is simple. A graph is a shorter description of the data whenever many records share a pattern that can be stated once and then pointed to, and a node earns its place when it saves more bits than it costs to state. Both sides of that comparison are bits, so there is no exchange rate to set. The price then comes out of the model exactly, with the constant on $\log_2 n$ equal to one (Lemma 2), and no quantity in it is calibrated on data.

That ought to be the end of the rule, and it is not, because the price cannot be paid at the moment it falls due. Ask what a brand-new node knows about the record that suggested it. It has no members and no counts, so its prediction over the values of a key is uniform, and under any valid code a uniform guess cannot beat what the stream already knows. Lemma 1 makes this exact: in expectation a fresh node loses bits on its own first record whatever code we choose, and seeding it with that record's value only moves the same bits into the seed. So the rule that creates a node the first time a record fails to fit is not a rule that a better code can repair.

What works instead is to wait. The bits a candidate node would have saved are credited to an account named by the (key, value) pair that opened it, and every later record that goes unexplained in the same way pays into that account. The node is created at the first moment the account covers the price, and not before (Figure 2). We call the method ESCROW, after the account.

Waiting raises a question of its own. A candidate that is tested again after every arrival, forever, has a great many chances to cross its price by luck alone. In principle the escrow answers it. The account is a running score of fair bets, and a classical inequality of Ville (1939) caps the chance that the wealth from such bets ever reaches a given level, over the whole infinite life of the stream, with no correction for how often we look. That is the guarantee we would like to claim, and not the one we can claim for our engine. The statistic the engine accrues is read off the records that carry the candidate's pattern, and that subsequence is chosen by the data, which is precisely what a fair game forbids; under the null the accumulator drifts. We report the measured exceedance in its place, and show that no charge added to the price repairs it (Theorem 1).

Contributions. Pricing a component in bits, codelength pruning of trees, computed charges inside boosting and attribute-by-attribute growth from a stream are prior work (Section 6) and are not claimed. What is claimed:

- **A prequential code for an unsupervised, multi-membership, one-pass stream.** A record may activate several nodes, no record carries a label, and each decision is taken once and cannot be withdrawn. That combination removes the two properties the codelength criteria we survey rely on, a conditional code that transmits only a label and additivity over a partition. The code given here requires neither.
- **A computed mint price, and a lemma that rules out immediate minting.** A Bayes-exact mint price with constant exactly one on $\log_2 n$, and the lemma that no valid code lets a node pay on its first record, which is what makes deferral necessary rather than convenient. The lifetime false-mint bound holds for a normalised idealisation; for the statistic the engine ships we report the measured exceedance instead of claiming the bound (Section 4). Neither has a significance level or a grace period.
- **An evaluation stated as a set of falsification tests.** Planted structure is recovered exactly in all 20 arrival orders, no node is created on 48 null streams where streaming trees grow to a mean of 91, and the creation-bias grid stays flat at the truth. Section 5 then puts the method against

	A. Distance rule, hand-set cut-off	B. Bayesian rule, small-variance limit	C. Language model pipeline	D. ESCROW: a price in bits
who decides a node exists	a distance below the chosen cut-off	a fit gain above the constant lambda	a sampled answer from the model	the data: saved bits cover a computed price
what a person must set	a number somebody chose	a number somebody chose (lambda)	a prompt, a temperature and a retrieval depth	nothing is calibrated: both sides are bits
same output on rerun	yes, once the number is fixed	yes, once lambda is fixed	no: 24, 78, 28 nodes on identical records; agreement 0.1647	yes: 6 nodes, identical on all 3 runs
a receipt per node	no, only a distance	no, only a penalty	no; and one catch-all node holds up to 85% of records	yes, in bits, for every node
cost per 1,000 records	not run on this stream	not run on this stream	107,512 to 157,821 tokens; 27 to 31 min on one A100	0 tokens; 1.5 s on one CPU core

Figure 1: Four ways to decide that a node exists, one per column. ESCROW (D, gold) is the only one whose price is computed. Distance rules (A) and small-variance Bayesian rules (B) answer with a number somebody chose. A language-model pipeline (C) answers by sampling: on 1,000 Lazada listings in one fixed order, three seeds returned 24, 78 and 28 nodes at 107,512 to 157,821 tokens per run, where ESCROW returned the same 6 nodes in about 1.5 s, with no tokens and a receipt per node (results/llm_graph_formation.json; Section 5.4).

oracle-tuned baselines, against a language model building the same graph on the same records, and on two public benchmarks that it fails.

2 Problem statement and core claim

Setting. A record r_n is the n-th item to arrive, and it is a small set of attribute key and value pairs such as `fabric = cotton`. We call one (record, key) slot a facet, and version 1 handles categorical values, meaning symbols drawn from a finite alphabet. A node is a latent group of records that share a pattern of keys and values, and the keys it models are its support. One node is not latent: the background is the model of an ordinary record, it owns whatever no other node explains, and every claim of structure has to beat it. After $n-1$ records the state is K nodes plus that background, each node carrying its members, its support and the counts of the keys and values its members published. A record may belong to several nodes at once. The stream is seen once and every decision is final.

The question. Every arrival puts two questions to every facet. Which existing node owns it? And if no node explains it, has it now earned a new node, and at which record? A rule that answers with a chosen constant has a calibrated parameter. This paper asks whether the rule can have none (Figure 1).

The claim. *A node is created at the first moment the bits the stream has saved by positing it cover the computed bits it costs to state it. Both sides are bits, so no quantity in the rule is calibrated on data, and the same code decides which node each record joins.*

What is claimed and what is not. We claim that no parameter is calibrated. We do not claim the absence of bias, because Kearns et al. (1997) prove that for every sample size there are instances on which any penalty-based rule stays a fixed distance from the best answer, so we measure our creation bias directly instead (E13). What is new is a code that works in a setting where the existing codelength criteria cannot, rather than a new principle of model selection. Three concessions come with the claim. The one declared assumption is the starting prior on the stream's own minting rate, and it is worth 0.19 bits per mint at one hundred thousand records on a declared trajectory, 0.34 on the one our run walks. The candidate pool is held to a declared operational budget, which is not part of the criterion but can change the output (Section 7). And version 1 works on discrete facets only (Section 7).

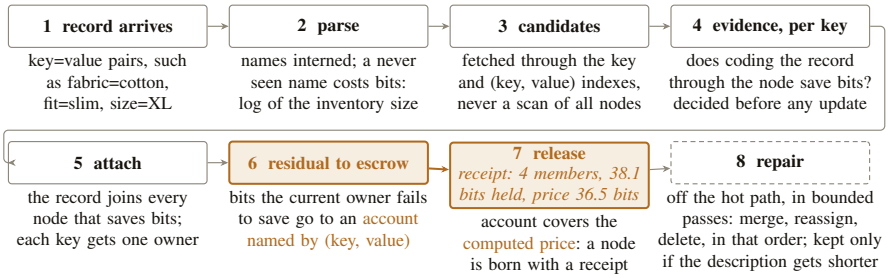


Figure 2: One record through ESCROW. Steps 1 to 5 and 8 are bookkeeping in bits, and the two gold stages are where a node earns existence. Bits the current owner fails to save are credited to an escrow account named by the (key, value) pair that carried them, and when the account covers the price a node is minted with a receipt, here 4 members and 38.1 bits held against a price of 36.5 bits on the synthetic two-group fixture (results/e9_support_curve.json).

3 THE METHOD, ESCROW

The code and its terms. The code is prequential, which means every record is priced by a prediction made from the past alone (Dawid, 1984), and every term in it satisfies Kraft's inequality, the rule that codeword lengths ℓ are realisable exactly when $\sum 2^{-\ell} \leq 1$. A node states two things about itself. The first is its member set, a yes-or-no column over the records so far, coded by the Krichevsky-Trofimov code, KT, which is the standard code for symbols of unknown frequency (Krichevsky & Trofimov, 1981; Willems et al., 1995). That column is the term which replaces λ, and nothing in it is chosen. The second is how reliably its members publish each key in its support, so that a member which stays silent on a supported key pays a silence bill. The bill matters: when two planted groups share no keys at all, it is the only signal there is.

Multi-membership brings a bookkeeping problem with it. A record may belong to several nodes, and if two of them claimed the same facet its bits would be counted twice. So every facet has exactly one owner, the node with the strongest claim on it, and the winning facet is emitted as a typed edge from the (key, value) pair to that owner, labelled with the key that decided it and the bits it paid. The test suite asserts that every (record, key) cell is coded exactly once by exactly one block (`code/tests/test_batch.py`). Additivity is bought by single ownership, and that is a circularity we state rather than smooth over.

The price, and the origin of the classical constant. The model the code transmits is the Indian buffet process, IBP, a Bayesian model of a stream in which a new feature may appear at any time (Griffiths & Ghahramani, 2011). Under it, record n joins an existing node k with probability m_k/n, where m_k counts the earlier members of k, and opens new nodes at rate γ/n. We do not fix γ: it gets a prior and is integrated out. What is left is a seed-time price for a candidate first proposed at record s, and it depends on two global counters, K_{s-1}, the nodes minted so far, and the harmonic sum $H_{s-1} = 1 + \frac{1}{2} + \cdots + \frac{1}{s-1}$, the accumulated opportunity to mint.

$$\Lambda_s \;=\; \log_2\big(s\,(b + H_{s-1}) + 1\big) - \log_2\big(a + K_{s-1}\big) \;=\; \log_2 s \;-\; \log_2 \hat{\gamma}_{s-1} \;+\; o(1), \qquad (1)$$

where $\hat{\gamma}_{s-1} = (a + K_{s-1})/(b + H_{s-1})$ is the stream's own running estimate of its minting rate, and the constant on $\log_2 s$ is exactly one. The pair $(a, b) = (\frac{1}{2}, 1)$ is a declared prior on that rate, Jeffreys in shape and forced to be proper, of finite total mass, by Kraft's inequality. Another reasonable proper prior moves one mint price by 0.19 bits at one hundred thousand records on a declared trajectory and 0.34 on the one our own run walks, and $\Lambda_1 = 2$ bits. BP-means is the small-variance limit of this same model (Broderick et al., 2013b), and that limit is where the computable parts of the price are lost (Section 4).

Why immediate minting fails. Lemma 1 states exactly what Section 1 argued in words: a brand-new node predicts uniformly, an uninformed guess cannot beat what the stream already knows against a strictly positive price, and seeding the node at the record's own value only relocates the same bits into the seed. E7 (Section 5.2) measures both immediate forms at zero mints, in all twenty cells of an alphabet by typo-rate sweep, so under a computed price the failure of the myopic rule is

silence. A constant DP-means-style price fails in the other direction, minting on rare values, which in practice means minting on typos, and E7 measures that arm too.

Escrow and release. Each key the graph fails to explain leaves a residual, in bits, and that residual accrues in an account held against a candidate node that does not exist yet. The candidate is named by its seed pair (key, value), one candidate per residual pair, and every later record whose key goes unexplained joins its cohort. The account holds the bits the incumbent owners actually spent on that cohort, minus the bits the candidate's own growing blocks would have spent. The node is created at the first time t^* at which some support T, a set of its keys, satisfies

$$G_t(T) \;\geq\; \mathrm{Price}(t, T, n), \tag{2}$$

where $G_t(T)$ is the evidence escrowed on the keys of T and $\mathrm{Price}(t, T, n)$ is the cost of stating the node itself: its announcement, its membership column, its support and its name (Equation D.10). The shipped gate charges Price alone, and the statistic released is the positive-part prefix over the candidate's keys sorted by escrowed bits plus the positive absence credits earned outside the support, so the statement terms can only delay a release and never license one (Appendix D). Under the null 2^{G_t} behaves like the wealth of a gambler in a fair game, which is Theorem 1. Across nine valid codes the minted set is identical on planted structure and moves where the structure is not identifiable, by no more than a change of arrival order does on one stream and by more on the other (E12).

Per-record processing, and repair. Per record the engine performs the eight steps of Figure 2, drawn as a flowchart in Figure 6 (Appendix D.1). Every statistic is snapshotted before any update, so every bet is priced fairly, and candidates are retrieved by index, which keeps the cost per record flat in the node count when supports are disjoint and does not when they all overlap (Section 7). The one-pass rule is greedy and irrevocable, so a bounded repair pass runs off the hot path. It descends the batch codelength, the cost of the state plus all records given it, which is a function of counts alone, using merge, reassign and delete moves that are accepted exactly when they strictly shorten the description, and it therefore terminates (Proposition C.4). The pass runs on a fixed record cadence, which belongs to the run protocol (Section 5.1) and not to the criterion.

Keys. Real streams drift, so `distributor` turns up beside `distributors`. Exact string matching cannot see that, and a learned embedding can, but then the model decides without leaving an audit trail. In version 1 a never-seen key opens blocks of its own and enters a node's support only when a node is minted on it. The operator that would price two keys as one codes the choice between the existing keys and NEW by the bits its records cost either way; it is designed but not run in this version (Section 7).

4 THEORETICAL GUARANTEES

Logarithms are base 2, so every quantity is in bits; each result gets one plain sentence, its formal statement and a proof sketch, and Appendix C restates and proves every result.

The origin of the classical constant. Before the small-variance limit is taken, the BP-means penalty is a codelength in disguise. Under the mass scaling of Broderick et al. (2013b) it satisfies $\lambda^2 = 2\sigma^2 \cdot (-\ln \gamma)$, twice the assumed noise variance σ^2 times the prior codelength of one new node, and the negative log joint probability carries three new-node terms: a mass term, a parameter cost and an allocation cost. Multiplying by $2\sigma^2$ and letting $\sigma \to 0$ annihilates the two computable costs and keeps the mass term alone (Lemma C.1, Appendix C, with the DP-means corroboration of Kulis & Jordan (2012)).

A fresh node cannot pay on its first record. Write $D(p\|q)$ for the Kullback-Leibler divergence, the expected extra bits paid for coding data from p with a code built for q, zero iff $p = q$; write $\hat{P}_{0,a}$ for the background's prediction on key a, whose alphabet has d_a symbols.

Lemma 1 (myopic impossibility)**.** *Let Q be the prediction a node minted at record n assigns to a value of key a before any record has attached to it. (i) In a valid code Q depends on the transmitted past alone and sums to at most one; with no past, exchangeability over values, treating every value alike, forces Q uniform, so every value costs $\log d_a$, and seeding Q at the observed value relocates the same $\log d_a$ bits into the seed. (ii) With p the law of the value, the expected first-record*

evidence against the background is $-D(p\|Q) + D(p\|\hat{P}_{0,a})$, which under the null $p = \hat{P}_{0,a}$ equals $-D(\hat{P}_{0,a}\|Q) \le 0$, while (iii) $\Lambda_n > 0$ whenever $a + K_{n-1} < n(b + H_{n-1}) + 1$. So in expectation a fresh node minted at record n loses at least Λ_n bits on every key of its first record (the pointwise exceptions, values the background prices below $2^{-\Lambda_n}/d_a$, are in the appendix restatement).

Proof sketch. (i) is Kraft's inequality plus exchangeability, the KT code at zero counts realising the uniform exactly; (ii) is the identity $\mathbb{E}_{x\sim p}[\log Q(x) - \log \hat{P}_{0,a}(x)] = -D(p\|Q) + D(p\|\hat{P}_{0,a})$ at $p = \hat{P}_{0,a}$; (iii) is $2^{-\Lambda_n} < 1$ under the stated condition, with $\Lambda_1 = 2$ at the declared prior. Full proof in Appendix C. □

The price is exact. The creation rate is given a Gamma prior, the standard prior for a positive rate, and is then integrated out. What is left is the log posterior odds of no mint against one mint, that is, how much more probable no mint is once the data are in, measured in bits. NML is normalized maximum likelihood, a prior-free code which exists only when its normaliser, the sum that turns its scores into probabilities, is finite.

Lemma 2 (the price)**.** *Under the IBP with $\gamma \sim \mathrm{Gamma}(a, b)$ marginalised, the log posterior odds of no new node against one new node at record s is exactly Λ_s of Equation 1, with remainder $O(1/(s \log s))$, and attachment odds contain no γ. Moreover: (i) the prior must be proper, because the NML normaliser over the node count, $\sum_K K^K e^{-K}/K!$, diverges, and an improper prior is not a code; (ii) two proper priors differ per mint by $O(1/\log n)$: 0.69 bits at $n = 10$ and 0.19 at $n = 10^5$ along the trajectory $K_s = 2H_s$, and 2.36 and 0.34 along the one our planted run walks; (iii) learning γ inside the code costs $\frac{1}{2}\log\ln N + O(1)$ bits over the whole stream.*

Proof sketch. The IBP allocation probability is $W(Z)\,\gamma^{K_N} e^{-\gamma H_N}$ with W free of γ (Griffiths & Ghahramani, 2011), so the posterior before record s is $\mathrm{Gamma}(a + K_{s-1},\ b + H_{s-1})$ and the odds of one new node against none are $(a + K_{s-1})/(s(b + H_{s-1}) + 1)$, whose negative base-2 logarithm is Λ_s; (i) to (iii) are proved in Appendix C. □

The lifetime false-mint bound holds in the idealised setting; for the engine we report the measured exceedance instead of claiming it. A martingale is a fair game, a process whose expected next value is its current one. Ville's inequality caps at $1/c$ the chance that a nonnegative martingale started at one ever exceeds c, over its whole infinite future (Ville, 1939; Ramdas et al., 2023).

Theorem 1 (anytime validity of the escrow)**.** *Fix a seed pair $p = (a^\dagger, v^\dagger)$ with seed time s and frozen price Λ_s, and let $G_t(p)$ be the accumulated log-likelihood ratio from s onward, over the whole stream, between the prequential graph with p and the current graph without it. (i) Under the null $H_0(p)$, that the without-p model generates the stream, $M_t = 2^{G_t(p)}$ is a nonnegative martingale with $M_s = 1$, exactly when every value block is exactly normalised, which a tight naming charge for the escape achieves and the shipped charge does not; the shipped code remains a valid prefix code, satisfying Kraft with inequality rather than equality, and the one-step expectation it measures is 0.862862 against the tight charge's 1.000000 (Appendix C). (ii) In that idealised setting $\mathbb{P}(\exists\, t \ge s : G_t(p) \ge \Lambda_s) \le 2^{-\Lambda_s} < \hat{\gamma}_{s-1}/s$, with no union bound over time (no summing of error over the times tested); the shipped gate, which releases at the price of Equation D.10 alone, is covered by the same inequality at the level $c = \min_t \mathrm{Price}(t, T, n)$. We state (ii) for the idealised setting and do not claim it for the engine. (iii) The hypotheses are indexed by seed pairs, one per residual (key, value) pair, a set that grows with the schema and not with the record count; the price names the winner, and because $\sum 2^{-\mathrm{Price}} \le 1$ (gate UT-4) that charge is the multiple-comparisons correction, so the expected number of spurious mints over the whole lifetime is controlled by the Kraft sum and not by the record count, whereas under mint-on-first-miss the hypotheses are one per record and the bound grows as $\Theta(T)$ unless the charge knows the stream length. It corrects multiplicity and multiplicity alone, and is therefore not available as a repair for (ii).*

Proof sketch. (i) $M_t = \prod_{u=s}^{t} Q_u(x_u)/P_u(x_u)$ with Q_u, P_u the one-step predictives, the probabilities each model assigns to the next record from the past alone (Dawid, 1984); under H_0 the ratio cancels the true law once both are normalised, which is the naming charge of (i). (ii) Ville at $c = 2^{\Lambda_s}$ with Lemma 2, and at $c = 2^{\min_t \mathrm{Price}}$ for the shipped gate. (iii) The support and name terms of the price are Kraft-tight (gate UT-4), so the sum over candidates is bounded by the Kraft sum. Two

caveats, stated in Appendix C: the implemented escrow drops absence terms that can only inflate G_t, and their bound is charged back through the membership column of Equation D.10; and the released statistic is the positive-part prefix plus outside absence credit of Section 3, computed on the records that carry the seed pattern, a subsequence the data chose, which is why (ii) is not claimed for the engine. □

What the engine measures instead of the bound, and why the price cannot repair it. Over 300 null streams and 12,000 candidate trajectories the exceedance $\Pr(\sup_t G_t \geq b)$ is 1.000 at every $b \in \{1, 2, 4, 8, 12\}$ for the shipped statistic, against a nominal 2^{-b}. Adding the seed naming charge of (iii) to the price, which is the repair the theorem itself contemplates, does not move that number in any of three codes. The reason is not multiplicity. Under the null the accumulator drifts, so its supremum grows with the stream: the slack that would restore the level at $b = 8$ grows from 97 bits at $T = 500$ to 1,205 at $T = 4{,}000$, while a naming charge is constant in T by construction. Remove the seed key's own term and the requirement becomes 3.4 to 3.6 bits and stays flat, and the same charge then covers it at all five levels, but that arm is not shippable because removing the term destroys the mint. The failure is a drift induced by the records the evidence is read on, and not a family-wise error rate (Appendix C).

Further results are stated and proved in Appendix C: exactness of the per-key receipts under the code the engine ships (Lemma C.2); the escrow as a meter of total correlation (Proposition C.2); consistency with no significance level (Proposition C.3); the closed-form overlap with structural entropy (Proposition C.1); and termination of the repair pass on an order-invariant batch codelength (Proposition C.4).

5 EXPERIMENTS

5.1 EXPERIMENTAL DESIGN

No price, cut-off or level is set per run, and every input ESCROW receives, every baseline receives. Table 5 states the question, design, control, baselines, tuning, metric and falsifier for each experiment, and Appendix G.2 states the run protocol and defines the *oracle*, *transferred* and *handed the true K* conditions. An audit of our implementation before these runs found three defects. The worst coded some cells twice, which made the batch objective depend on more than the graph. Every number below comes from the corrected engine, and Appendix E holds the guarding tests.

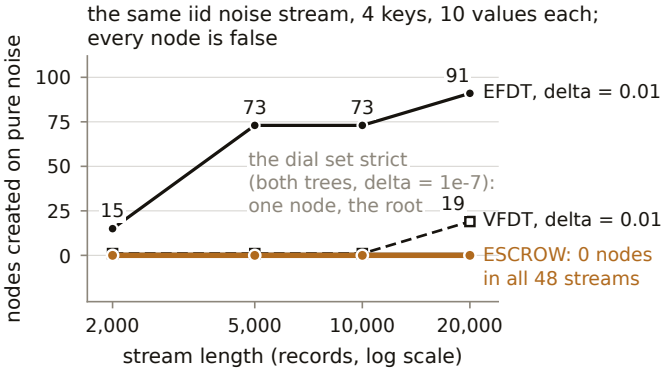
5.2 MECHANISM FALSIFICATIONS

E7: immediate mint is broken in both forms; deferral is not. Both immediate arms mint zero times, which is what Lemma 1 says they must do. The record-seeded arm does it in all twenty cells of an alphabet by typo-rate sweep, because a whole typo record is worth 23.1 bits against a node price of 39.8. A constant price fails in the other direction. At sixteen bits every node it mints is seeded by a typo and then absorbs ordinary records, so the corruption becomes permanent structure. The deferred arm ends at $K = 2$ on all five seeds, with exactly the two planted key sets as its supports, and cites no typo value at any mint (Appendix G.3).

E8: no node is created on noise, and what the bound covers. Table 1 shows the Ville bound holding with slack at every level on a standalone one-key block. That block is not the engine's accumulator, whose measured exceedance is the subject of Theorem 1. The engine's own result is the practical one. K ends at 0 on 48 of 48 null streams, flat from 2,000 to 20,000 records, and on 300 further streams, with no significance level and no grace period anywhere in the rule. The streaming trees do not stay flat (Figure 3), and the strict setting that silences them costs 20,800 records of delay when structure is present and never splits at higher noise (Appendix G.5).

E9, E13 and E5: the price behaves like a price, the bias is not the one Kearns predicts, and the order cost is measured. A price should ask for more evidence when each record carries less. In all 12 cells of the (k, d) grid the cohort size at birth falls as k grows and never rises, and the overshoot at the mint is below one member's average evidence in every cell (Table 8).

The creation bias runs the opposite way from the one Kearns et al. (1997) prove. On a grid of 270 runs whose fixture can fail in either direction, six noise levels by three lengths by three planted



b (bits)	2^{-b}	empirical
1	0.5	0.09785
2	0.25	0.0324
4	0.0625	0.0064
8	0.0039	0.0002
12	0.00024	0.0000

Table 1: E8, Ville bound: empirical $\Pr(\sup_t G_t \geq b)$ over 20,000 null trajectories of 200 steps on a standalone one-key block, below 2^{-b} at every level.

Figure 3: No node is created on noise. Every method sees the same pure-noise generator, so every node created is false. ESCROW (gold) creates zero nodes at every length in all 48 streams. The streaming trees VFDT and EFDT at $\delta = 0.01$ grow with the stream, five seeds per length, EFDT to means of 15, 73, 73 and 91 nodes (maximum 111) and VFDT to 19. At $\delta = 10^{-7}$ both stay at the root (results/e8_false_mint_null.json, results/e8_tree_arms.json).

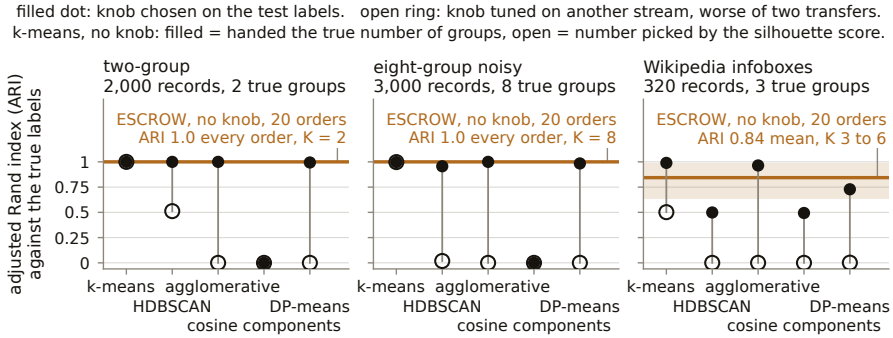


Figure 4: Tuned baselines against ESCROW on three streams, scored by ARI. Filled markers are oracle scores, with the knob chosen on the test labels, and open markers are transferred scores, with the knob tuned on another stream, the worse of two transfers. The grey line is the drop between them: to 0.0 for agglomerative and DP-means on the synthetic streams, and to at most 0.0017 for every transferred knob on Wikipedia. k-means has no knob, so filled is handed the true K and open picks K by silhouette. ESCROW (gold) has nothing calibrated and no embedding (results/e4_first_tier.json, results/e5_order_dependence.json).

counts at five seeds, no run ever returned more nodes than were planted and no row grows with length. Ten rows converge to K^* and eight plateau below it, so the method writes structure off past a breakdown noise of 0.2, 0.3 and 0.4 at two, eight and twenty planted groups (Table 9).

The cost of arrival order is measured rather than assumed. The batch objective is a function of the state and not of the order, which the test suite asserts as invariance to relabelling, but the greedy sequence that reaches those states is not. Over 20 random orders both synthetic streams return ARI 1.0 and the exact K every time, while Wikipedia returns 0.8438 on average and ranges from 0.6389 to 0.986 with K from 3 to 6. That range is ESCROW's Wikipedia result, and the single values quoted below are instances drawn from it (Table 10).

5.3 Tuned baselines against no knob

Figure 4 holds three facts. ESCROW recovers both synthetic streams exactly with nothing tuned. Transferred knobs collapse, and the one label-free self-tuner fails with them, where ESCROW has no knob to transfer in the first place. On Wikipedia its mean of 0.84 over 20 orders sits above every transferred knob and below the two baselines allowed to touch the labels, each a single run on a fixed embedding (Table 11).

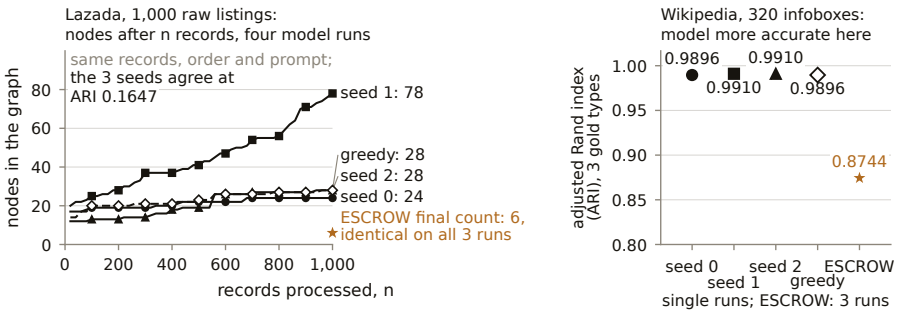


Figure 5: A language model building the same graph on identical records. Left: the first 1,000 raw Lazada listings. Four runs of Qwen2.5-14B-Instruct saw the same records, order and prompt, yet end with 24, 78, 28 and 28 nodes, agreeing at a mean pairwise ARI of 0.1647; the gold star is ESCROW's 6 nodes on all three runs, with 699 records left in its background. Right: Wikipedia on one arrival order, where the model is more accurate (results/llm_graph_formation.json, results/llm_gf_runs/).

5.4 A LANGUAGE MODEL BUILDING THE SAME GRAPH

Qwen2.5-14B-Instruct (Qwen Team et al., 2024) is prompted to build the graph as a stream, three seeds at temperature 0.7 and one greedy run, against ESCROW on the identical records in the identical order (Figure 5; the prompt protocol is Table 5). On Wikipedia, which is 320 clean records of three obvious types, the model wins. It reaches ARI 0.9905 and purity 1.0 at $K = 4$, against ESCROW's 0.8744 and $K = 5$ on this arrival order, and that order is one draw from the spread of 0.6389 to 0.986 which E5 measures over twenty, so the model is ahead of our mean and inside our range.

On 1,000 raw Lazada listings the measurement is determinism and cost rather than accuracy. The silver truth carries 759 types over 1,000 records and 614 of them are singletons, so ARI is uninformative and near zero for every method, so the failure modes are what we measure. The three seeded runs agree with one another at a mean pairwise ARI of 0.1647, where ESCROW agrees with itself at 1.0. A catch-all node holds 23 to 85 percent of the records, and in one seed 77 percent of the nodes are named after a value copied out of a record (Appendix G.15). With no rule for when a node should exist, the graph comes out non-deterministic, catch-all and value-named at scale. ESCROW returns the same six nodes on every run with a receipt for each, and leaves 699 records in its background. Figure 8 gives the bill in tokens and in GPU-hours.

5.5 REAL-WORLD STANDING

Four public streams, one arrival order each (Table 12; the runs in detail are Appendices G.13, G.14, G.16 and G.17). **Wikipedia** settles at $K = 5$ with purity 0.9665, and its large types split differently across orders, which is why K ranges from 3 to 6 (E5). **Lazada**, 21,365 listings over 3,033 raw keys, settles at $K = 68$, where the largest nodes cross-list specification templates, and case drift is a stated negative. **MusicBrainz 20K** is the sharpest negative. Its five categorical fields carry no identity signal for anyone, so condition A is null for every method. Under the token bridge of condition B, where three of four oracle-tuned embedding baselines reach 0.61 to 0.78 pairwise F1, ESCROW reaches 0.0011. Identity there is a title one token apart, which a categorical facet cannot see and an embedding sees at once. **ReVerb45K** states its limit first, a macro F1 of 0.016, because 7,009 subjects end as singletons and most gold clusters are never formed. Its micro and pairwise F1 of 0.7777 and 0.7160 sit just below the benchmark's no-merge floor, which is what the embedding baselines reach by returning close to one cluster per surface string.

6 RELATED WORK

Appendix B is the full survey, with every number and its source. Pricing a component in bits is Wallace & Boulton (1968); codelength pruning of trees is Quinlan & Rivest (1989), Wallace & Patrick (1993) and Mehta et al. (1995; 1996), each relocating a knob rather than removing it; Kearns

et al. (1997) prove that no reweighting of a two-part code removes its bias, accepted here as scope and measured (E13). Computed charges exist in batch, inside boosting (Friedman, 2001; Chen & Guestrin, 2016; Ke et al., 2017; Lunde et al., 2020) and across a description-length literature in which parameter-free is already a title (Akoglu et al., 2012; Koutra et al., 2014; Vreeken et al., 2011; Miettinen & Vreeken, 2014; Zanella et al., 2016; Mahon & Lapata, 2025; Peixoto, 2017; Yang & van Leeuwen, 2024; Galbrun, 2022), each selecting structure over data already in hand. Streaming growth, VFDT and EFDT (Domingos & Hulten, 2000; Manapragada et al., 2018) and their anytime-valid repair (Amoukou et al., 2026; Ville, 1939; Ramdas et al., 2023), is supervised and uses no description length. Structural entropy (Li & Pan, 2016; Cao et al., 2024; Xie et al., 2025) is the nearest unsupervised lineage and does construct, but its graph forgets which key typed a value (Proposition C.1). Language-model pipelines (Bai et al., 2025; Hongwimol et al., 2026) already ask whether a key should be added, merged or discarded, by a prompted model with no edit carrying its evidence; the question is theirs, the criterion ours (Section 5.4). Per-variable decompositions of information scores exist (Silander et al., 2018; Kontkanen et al., 2003; Costa et al., 2026; Slonim & Tishby, 1999; Wallace & Dowe, 1994); new is their job as the decision variable of an online create test.

7 LIMITATIONS

Appendix G.19 states all of this at length, and the external dataset tier (Appendix G.18) is not run. **Categorical facets only.** Free text is quarantined, because any code for it must first choose a tokenisation, which we measure at 5 to 75 bits per field, more than any constant the method removes. On free-text identity version 1 leaves most subjects as singletons, and Section 5.5 reports how far that falls short of the tuned embedding baselines. The numeric code is quarantined until it has been evaluated, and the operator that prices two keys as one is designed rather than run. **The search is approximate, and its budget is measured rather than tuned.** The exact argmax is a maximum edge biclique problem, which is NP-hard (Peeters, 2003), so the criterion is evaluated exactly over a pool held to a declared operational budget, and eviction drops the account with the lowest accrued evidence. E19 sweeps that budget over sixteen values from 1 to 32,768 on four streams at five seeds each. The output stops moving at 128 on the two-group stream, 256 on the planted eight-group stream, 2,048 on Wikipedia and 3,072 on the Lazada first 1,000, and is identical at every larger budget, so the shipped default of 4,096 sits on the flat part of all four curves. The plateau is not itself a tuned value. On each stream it is the point where the pool stops evicting, fixed by the peak number of live candidate accounts that stream produces (74, 468, 1,921 and 2,378), so the budget sits above a property of the stream rather than being searched for. Below the plateau it does change the output, and not monotonically, which makes it a resource bound rather than a quality knob: from 128 upwards it moves the Wikipedia index by 0.139, where the arrival order moves it by 0.320. These streams run 320 to 3,000 records, and the cap binds on the full Lazada stream and on MusicBrainz 20K, so the claim is made at that scale. **Cost.** The insertion step is flat in the node count when supports are disjoint, at 48.6 microseconds per record for 8 nodes and 45.0 for 128, and it is not flat when they all overlap. End-to-end throughput falls from 7,892 to 909 records per second between 10^3 and 10^5 records, because at the top the repair pass takes 95.8 percent of the time, and that is the mechanism behind the twentyfold spread between our real-world runs. **Order dependence.** The objective is a function of the state, and the greedy pass that reaches states is not, so E5 measures the spread it produces (Section 5.2).

8 CONCLUSION

The classical rules price a new node with a constant that a limit left behind, after erasing every part of the price a model could have computed for itself. We did not take that limit. We coded the stream prequentially under the Indian buffet process and read the price off the code, and since no fresh node can pay on its own first record, the evidence waits in escrow until an account covers a price. Numeric and text facets, key identity and the head to head on the language-model system's own data come next.

REFERENCES

Leman Akoglu, Hanghang Tong, Brendan Meeder, and Christos Faloutsos. PICS: Parameter-free identification of cohesive subgroups in large attributed graphs. In *Proceedings of the 2012 SIAM International Conference on Data Mining (SDM)*, pp. 439-450, 2012.

Salim I. Amoukou, Saumitra Mishra, and Manuela Veloso. Correcting split selection in online decision trees via anytime-valid inference. *arXiv preprint arXiv:2605.31239*, 2026. Accepted as a spotlight at ICML 2026.

Jiaxin Bai, Wei Fan, Qi Hu, Qing Zong, Chunyang Li, Hong Ting Tsang, Hongyu Luo, Yauwai Yim, Haoyu Huang, Xiao Zhou, Feng Qin, Tianshi Zheng, Xi Peng, Xin Yao, Huiwen Yang, Leijie Wu, Yi Ji, Gong Zhang, Renhai Chen, and Yangqiu Song. AutoSchemaKG: Autonomous knowledge graph construction through dynamic schema induction from web-scale corpora. *arXiv preprint arXiv:2505.23628*, 2025.

José M. Bernardo and Adrian F. M. Smith. *Bayesian Theory*. Wiley, 1994.

Tamara Broderick, Michael I. Jordan, and Jim Pitman. Cluster and feature modeling from combinatorial stochastic processes. *Statistical Science*, 28(3):289-312, 2013a.

Tamara Broderick, Brian Kulis, and Michael I. Jordan. MAD-Bayes: MAP-based asymptotic derivations from Bayes. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pp. 226-234, 2013b. arXiv:1212.2126.

Yuwei Cao, Hao Peng, Zhengtao Yu, and Philip S. Yu. Hierarchical and incremental structural entropy minimization for unsupervised social event detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 8255-8264, 2024. doi: 10.1609/aaai.v38i8.28666. arXiv:2312.11891.

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 785-794, 2016. arXiv:1603.02754.

Bertrand S. Clarke and Andrew R. Barron. Jeffreys' prior is asymptotically least favorable under entropy risk. *Journal of Statistical Planning and Inference*, 41(1):37-60, 1994.

John G. Cleary and Ian H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396-402, 1984.

Efthymios Costa, Ioanna Papatsouma, and Angelos Markos. A deterministic information bottleneck method for clustering mixed-type data. *Pattern Recognition*, 179:113580, 2026. doi: 10.1016/j.patcog.2026.113580. arXiv:2407.03389.

Valter Crescenzi, Andrea De Angelis, Donatella Firmani, Maurizio Mazzei, Paolo Merialdo, Federico Piai, and Divesh Srivastava. Alaska: A flexible benchmark for data integration tasks. *arXiv preprint arXiv:2101.11259*, 2021.

A. Philip Dawid. Present position and potential developments: Some personal views. Statistical theory: The prequential approach. *Journal of the Royal Statistical Society, Series A (General)*, 147(2):278-292, 1984.

Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 71-80, 2000.

Finale Doshi-Velez and Sinead A. Williamson. Restricted Indian buffet processes. *arXiv preprint arXiv:1508.06303*, 2015.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189-1232, 2001.

Esther Galbrun. The minimum description length principle for pattern mining: A survey. *Data Mining and Knowledge Discovery*, 36(5):1679-1727, 2022. doi: 10.1007/s10618-022-00846-z. Extended manuscript: arXiv:2007.14009.

Thomas L. Griffiths and Zoubin Ghahramani. The Indian buffet process: An introduction and review. *Journal of Machine Learning Research*, 12:1185-1224, 2011.

Peter D. Grünwald. *The Minimum Description Length Principle*. MIT Press, Cambridge, MA, 2007.

Christopher Harshaw, Moran Feldman, Justin Ward, and Amin Karbasi. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 2634-2643, 2019. arXiv:1904.09354.

Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2): 151-178, 1998.

Pollawat Hongwimol, Haoning Shang, Chutong Wang, Zhichao Wan, Yi Gao, Yuanming Li, Lin Gui, Wenhao Sun, and Cheng Yu. AutoPKG: An automated framework for dynamic e-commerce product-attribute knowledge graph construction. *arXiv preprint arXiv:2604.16950*, 2026. Accepted to Findings of ACL 2026.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pp. 3146-3154, 2017.

Michael Kearns, Yishay Mansour, Andrew Y. Ng, and Dana Ron. An experimental and theoretical comparison of model selection methods. *Machine Learning*, 27(1):7-50, 1997.

Ari Kobren, Nicholas Monath, Akshay Krishnamurthy, and Andrew McCallum. A hierarchical algorithm for extreme clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 255-264, 2017.

Petri Kontkanen and Petri Myllymäki. A linear-time algorithm for computing the multinomial stochastic complexity. *Information Processing Letters*, 103(6):227-233, 2007.

Petri Kontkanen, Wray L. Buntine, Petri Myllymäki, Jorma Rissanen, and Henry Tirri. Efficient computing of stochastic complexity. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics (AISTATS)*, pp. 171-178, 2003.

Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. VoG: Summarizing and understanding large graphs. In *Proceedings of the 2014 SIAM International Conference on Data Mining (SDM)*, 2014. arXiv:1406.3411.

Raphail E. Krichevsky and Victor K. Trofimov. The performance of universal encoding. *IEEE Transactions on Information Theory*, 27(2):199-207, 1981.

Brian Kulis and Michael I. Jordan. Revisiting k-means: New algorithms via Bayesian nonparametrics. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012. arXiv:1111.0352.

Angsheng Li and Yicheng Pan. Structural information and dynamical complexity of networks. *IEEE Transactions on Information Theory*, 62(6):3290-3339, 2016. doi: 10.1109/TIT.2016.2555904.

Jianhua Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145-151, 1991.

Berent Å. S. Lunde, Tore S. Kleppe, and Hans J. Skaug. An information criterion for automatic gradient tree boosting. *arXiv preprint arXiv:2008.05926*, 2020.

Louis Mahon and Mirella Lapata. K*-means: A parameter-free clustering algorithm. *arXiv preprint arXiv:2505.11904*, 2025.

Chaitanya Manapragada, Geoffrey I. Webb, and Mahsa Salehi. Extremely fast decision tree. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1953-1962, 2018. arXiv:1802.08780.

Manish Mehta, Jorma Rissanen, and Rakesh Agrawal. MDL-based decision tree pruning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, pp. 216-221, 1995.

Manish Mehta, Rakesh Agrawal, and Jorma Rissanen. SLIQ: A fast scalable classifier for data mining. In *Advances in Database Technology, Proceedings of the Fifth International Conference on Extending Database Technology (EDBT)*, pp. 18-32, 1996.

Pauli Miettinen and Jilles Vreeken. MDL4BMF: Minimum description length for Boolean matrix factorization. *ACM Transactions on Knowledge Discovery from Data*, 8(4), 2014. Article 18.

René Peeters. The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics*, 131(3):651-654, 2003.

Tiago P. Peixoto. Nonparametric Bayesian inference of the microcanonical stochastic block model. *Physical Review E*, 95:012317, 2017. arXiv:1610.02703.

J. Ross Quinlan and Ronald L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80(3):227-248, 1989.

Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Aaditya Ramdas, Peter Grünwald, Vladimir Vovk, and Glenn Shafer. Game-theoretic statistics and safe anytime-valid inference. *Statistical Science*, 38(4):576-601, 2023.

Jorma Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):416-431, 1983.

Alieh Saeedi, Eric Peukert, and Erhard Rahm. Comparative evaluation of distributed clustering schemes for multi-source entity resolution. In *Advances in Databases and Information Systems (ADBIS)*, volume 10509 of *Lecture Notes in Computer Science*, pp. 278-293, 2017.

Yuri M. Shtarkov. Universal sequential coding of single messages. *Problems of Information Transmission*, 23(3):175-186, 1987. Translated from Problemy Peredachi Informatsii 23(3):3-17.

Tomi Silander, Janne Leppä-aho, Elias Jääsaari, and Teemu Roos. Quotient normalized maximum likelihood criterion for learning Bayesian network structures. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 84 of *Proceedings of Machine Learning Research*, pp. 948-957, 2018.

Noam Slonim and Naftali Tishby. Agglomerative information bottleneck. In *Advances in Neural Information Processing Systems 12 (NIPS)*, 1999.

Shikhar Vashishth, Prince Jain, and Partha Talukdar. CESI: Canonicalizing open knowledge bases using embeddings and side information. In *Proceedings of the 2018 World Wide Web Conference (WWW)*, pp. 1317-1327, 2018.

Jean Ville. *Étude critique de la notion de collectif*. Gauthier-Villars, Paris, 1939.

Jilles Vreeken, Matthijs van Leeuwen, and Arno Siebes. Krimp: Mining itemsets that compress. *Data Mining and Knowledge Discovery*, 23(1):169-214, 2011.

Christopher S. Wallace and David M. Boulton. An information measure for classification. *The Computer Journal*, 11(2):185-194, 1968.

Christopher S. Wallace and David L. Dowe. Intrinsic classification by MML: The Snob program. In *Proceedings of the 7th Australian Joint Conference on Artificial Intelligence*, pp. 37-44. World Scientific, 1994.

Christopher S. Wallace and Jon D. Patrick. Coding decision trees. *Machine Learning*, 11(1):7-22, 1993.

Frans M. J. Willems. Coding for a binary independent piecewise-identically-distributed source. *IEEE Transactions on Information Theory*, 42(6):2210-2217, 1996.

Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41(3):653-664, 1995.

Qun Xie and Andrew R. Barron. Asymptotic minimax regret for data compression, gambling, and prediction. *IEEE Transactions on Information Theory*, 46(2):431-445, 2000.

Tianchi Xie, Jiangning Zhu, Guozu Ma, Minzhi Lin, Wei Chen, Weikai Yang, and Shixia Liu. Structural-entropy-based sample selection for efficient and effective learning. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.02268.

Lincen Yang and Matthijs van Leeuwen. Conditional density estimation with histogram trees. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024. arXiv:2410.11449.

Giacomo Zanella, Brenda Betancourt, Hanna Wallach, Jeffrey Miller, Abbas Zaidi, and Rebecca C. Steorts. Flexible models for microclustering with application to entity resolution. In *Advances in Neural Information Processing Systems 29 (NeurIPS)*, 2016. arXiv:1610.09780.

A TERMS USED IN THE APPENDIX

Terms used here: this section is itself the list. The appendices that follow use every term below without re-introducing it, and each opens with a two-line note naming the ones it leans on.

- **Record.** One item of the stream: a small set of attribute key and value pairs, such as `fabric = cotton`.
- **Facet.** One (record, key) slot: the place where one record gives one value for one key.
- **Node.** An entry in the graph that the system has decided exists: a latent group of records that share a pattern of keys and values.
- **Background node.** The model of an ordinary record: the node that owns whatever no other node explains, and the baseline every claim of structure must beat.
- **Support, scope.** The set of keys a node models. A node is charged for every key in its support and for nothing outside it. The proofs also say scope.
- **Typed edge.** An edge from a (key, value) pair to the node that owns it, labelled by the key that decided it and by its cost in bits.
- **Candidate, seed pair.** A node that does not yet exist, named by the (key, value) pair that first proposed it. That pair is its seed.
- **Cohort.** The records held in escrow behind one candidate: each record whose unexplained key carried the candidate's pair.
- **Null.** The hypothesis that a candidate is spurious: there is no structure behind it, and its evidence is noise.
- **Codelength, bits.** The number of bits needed to write something down under a probability model; a prediction that gives probability P to what happens costs $-\log_2 P$ bits, so a better model is a shorter description.
- **Prequential code.** A code that predicts each record before it is seen, from the past alone, and pays the bits of that prediction; nothing in it peeks at the future.
- **Batch codelength, L_{batch}.** The cost of writing down the current state and then all records given the state; a function of counts alone, and the objective the repair pass descends.
- **MDL.** The minimum description length principle: between two explanations of the same data, prefer the one that writes the data and the explanation down in fewer bits.
- **Two-part code.** A code that writes the model first and then the data given the model.

- **Kraft's inequality.** Codeword lengths ℓ can be realised by a real code exactly when $\sum 2^{-\ell} \leq 1$; equivalently, only a (sub)probability can be a code.
- **DP-means.** The k-means-like clustering rule derived from the Dirichlet process; it opens a new cluster when a record's distance to every centre exceeds a penalty λ.
- **BP-means.** Its cousin derived from the beta process; it lets a record belong to several features at once and opens a new feature at the same kind of penalty.
- **Small-variance limit, MAD-Bayes.** The derivation that produces DP-means and BP-means: put a Gaussian noise model with variance σ^2 on the features, scale the negative log joint by $2\sigma^2$, and let $\sigma \to 0$.
- **IBP.** The Indian buffet process: a Bayesian model of a stream in which each record activates a set of features and a new feature may appear at any time, with creation rate (concentration, mass) γ.
- **Hyperprior.** The prior placed on the model's own creation rate γ; here $\mathrm{Gamma}(a, b)$ with declared $(a, b) = (\frac{1}{2}, 1)$.
- **KT code.** The Krichevsky-Trofimov code: a standard way to price a sequence of symbols whose frequencies are unknown, by giving every symbol half a count up front.
- **NML.** Normalized maximum likelihood: the minimax code for a model class, which exists only when its normaliser, the Shtarkov sum of maximised likelihoods over all data, is finite.
- **Regret.** The extra bits a code pays over the best code chosen in hindsight.
- **Escrow.** A held account: evidence for a node that does not yet exist accumulates there, and the node is created only when the account covers the node's price.
- **Mint.** To create a node. A candidate is minted at the first moment its escrowed evidence covers its price.
- **Price.** The number of bits a new node must earn before it may exist: $\mathrm{Price}(t, T, n)$ of Equation D.10, the cost of stating the node, computed from the stream and never chosen. Λ_s is the seed-time term derived in the theory; it is not part of the shipped gate.
- **Martingale, Ville's inequality.** A martingale is the wealth of a gambler in a fair game: its expected next value equals its current value. Ville's inequality says a nonnegative martingale started at one ever exceeds c with probability at most $1/c$.
- **Kullback-Leibler divergence, $D(p\|q)$.** The expected extra bits paid for coding data from p with a code built for q; zero iff $p = q$.
- **Jensen-Shannon divergence.** The mutual information between one observation and which of two distributions produced it: how distinguishable the two are, in bits.
- **Total correlation.** The bits per record lost by modelling keys independently when a latent distinction couples them; zero exactly when the keys are independent given the node.
- **Structural entropy.** The bits a random walk on a graph needs once the graph is cut into communities; the nearest unsupervised incumbent objective.
- **Submodular.** A gain in which each further item helps less than the one before.
- **Repair.** A bounded pass, off the hot path, that revisits earlier decisions by merging, reassigning or deleting nodes whenever doing so shortens the total description.
- **Gate (UT-n).** A test that ships with the code and must pass for the build to be trusted; numbered as in the released specification.
- **ARI.** The adjusted Rand index: a chance-corrected clustering score in which 1.0 is a perfect match to the true groups and 0.0 is what a random grouping would score.
- **Oracle, transferred.** A baseline knob chosen on the test labels themselves; a knob tuned on another stream and carried over unchanged.

B Related work in full

Terms used here: MDL is the minimum description length principle, which prefers the explanation that writes data and model down in fewer bits; a two-part code writes the model first and the data given the model; a Hoeffding bound is a statistical guarantee on how far a sample average can stray

from its mean; structural entropy is the bits a random walk on a graph needs once the graph is cut into communities; ARI is the adjusted Rand index, a chance-corrected clustering score.

Codelengths for tree growth: the relocated parameter. MDL, the minimum description length principle, prefers the model that writes data and model down in fewer bits. Pricing a component in bits is Wallace & Boulton (1968): the component is created only when the bits it saves exceed the bits needed to state it. Quinlan & Rivest (1989) grew and pruned decision trees by one codelength, with no held-out set. But it relocated the parameter rather than removing it. The code carries a hand-set weight c on one of two bit counts. The paper sweeps c over 1, 2 and 8. That sweep moves one dataset from 17 to 69 leaves and from 20.5% to 13.0% error. The choice of c is left "as an open problem." Wallace & Patrick (1993) corrected the coding. But they introduced a Dirichlet concentration α, a knob that sets how spread out the assumed class frequencies are. They fitted it by a grow-and-refit loop, and it took values from 0.022 to 0.761 across datasets. Their formulation is ours. Adopting a code "is equivalent to accepting a certain prior probability distribution over the set of possible theories." Mehta et al. (1995) fix a precision knob at 1 after a sweep. SLIQ, the Supervised Learning In Quest decision-tree learner (Mehta et al., 1996), fixes an "empirically determined" test cost. It also records that the earlier codes over-pruned. Kearns et al. (1997) prove that no reweighting of a two-part code fixes its bias. A two-part code writes the model first and then the data given the model. For every sample size there are instances on which any penalty-based rule stays a fixed distance from the best answer. We accept that verdict as scope. We claim no calibrated parameter, not no bias. We measure the creation bias directly (Section 5).

Computed charges, in batch. A second lineage computes the charge, but with the whole dataset in hand. Boosting builds a model as a sum of small trees. Inside boosting the charge per new tree node is capped (Friedman, 2001), or tuned or absent (Chen & Guestrin, 2016; Ke et al., 2017). Lunde et al. (2020) derive that charge as an asymptotic optimism, an estimate of how much a split's gain overstates its true value for large samples. That leaves only the learning rate as a knob. In unsupervised learning, learning without labels, the phrase "parameter-free" is already widely used. PICS, Parameter-free Identification of Cohesive Subgroups (Akoglu et al., 2012), groups the nodes of a graph that carries node attributes by MDL, under exactly that title. VoG, Vocabulary-based summarisation of Graphs (Koutra et al., 2014), admits a subgraph exactly when it shortens the description. KRIMP (Vreeken et al., 2011), an MDL pattern miner whose name is the Dutch word for shrink, covers a transaction with several patterns. A transaction is one set of items, such as a shopping basket, and the patterns must split its items into non-overlapping parts. MDL4BMF, MDL for Boolean Matrix Factorisation (Miettinen & Vreeken, 2014), prices the Boolean rank of a record-by-attribute matrix of zeros and ones. Such a matrix is exactly an allocation matrix of the IBP, the Indian buffet process (Griffiths & Ghahramani, 2011). The IBP is a Bayesian model of a stream in which a new feature may appear at any time. But MDL4BMF needs the complete matrix. Microclustering (Zanella et al., 2016) derives the new-cluster cost that DP-means, the k-means-like clustering rule derived from the Dirichlet process (Kulis & Jordan, 2012; Broderick et al., 2013b), leaves tuned. It does so over a single number per record. k^*-means (Mahon & Lapata, 2025) "eliminates the need to set k", the cluster count, by MDL split and merge steps. Peixoto (2017) replaces the constant with hierarchies of priors. CDTree, a tree-shaped clustering method (Yang & van Leeuwen, 2024), says MDL "eliminates the need for tuning the hyperparameter". Its normaliser, the sum that turns its score into a probability, factorises only because each record reaches exactly one leaf. Galbrun (2022) names where such designs silently become parameter choices. A universal integer code, a fixed way to write any whole number in bits, is chosen "to favour partitions with fewer groups." Our answer is structural. The hyperprior, the prior placed on the model's own rate of creation, is forced proper by Kraft's inequality. Kraft's inequality is the rule that codelengths must correspond to probabilities summing to at most one. Proper means the prior's total mass is finite. It is not selected on data (Section 4). The encoding-stability sweep E12 measures the slack that remains (Appendix D). Every criterion above selects structure over data already in hand. The constant also survives into deployment: a deployed product-catalogue placement layer built on BP-means and DP-means that we examined carries five hand-set constants, calibrated by a sweep on 1,600 records, and its source comment concedes they must be re-tuned once the graph is live.

Streaming growth. Growing structure from a stream by picking the best attribute at each step is VFDT, the Very Fast Decision Tree (Domingos & Hulten, 2000). VFDT splits a leaf when a

Hoeffding bound separates the best attribute from the second best. A Hoeffding bound is a statistical guarantee on how far a sample average can stray from the truth. VFDT carries a significance level δ, a tie parameter and a grace period, a fixed number of records to wait before testing. EFDT, the Extremely Fast Decision Tree (Manapragada et al., 2018), splits as soon as the best attribute beats no split at all, and it can retract a split later. Amoukou et al. (2026) show the Hoeffding gate is invalid when the stopping time depends on the data. They repair it with an anytime-valid test-by-betting gate (Ville, 1939; Ramdas et al., 2023), a test that stays valid no matter when one stops looking. The repair keeps a significance level, a grace period of 20 records and an improvement threshold. It uses no description length. All of this is supervised, meaning every record must carry a label. Delete the label and the split gain is undefined. We are not aware of prior work that combines an irrevocable per-arrival node-birth decision as a codelength. Escrow, our held account of evidence for a node that does not yet exist, obtains lifetime validity from the encoding itself. It needs no significance level and no grace period (Theorem 1).

Structural entropy. The nearest unsupervised lineage, and it does construct. Structural entropy (Li & Pan, 2016) is the bits a random walk needs once a graph is organised into a hierarchy of communities. HISEvent, hierarchical and incremental structural entropy minimisation for event detection (Cao et al., 2024), adds meaning-based edges to a graph of messages by entropy minimisation before partitioning it. It needs no cluster count k. Four limits remain. Attribute identity is destroyed at construction. Two records share one unlabelled edge, weighted by the cosine similarity of their text embeddings. The node set is fixed at one node per message, so no creation decision exists. The parameter-free claim covers the cluster count only. Its own sensitivity study moves ARI, the adjusted Rand index, a chance-corrected clustering score, by ten points as the sub-graph size varies. And its "incremental" update runs over the neighbour rank, never over arriving records. SES, structural-entropy-based sample selection (Xie et al., 2025), decomposes structural entropy losslessly to nodes by a closed-form Shapley value, a fair split of a total score among the parts that produced it. So per-element decomposition exists in this line. But it is per node, never per attribute, because each record is first collapsed into one frozen embedding. And it runs under a user-set budget rather than a decision on whether structure should exist.

Pipelines built on large language models. AutoSchemaKG (Bai et al., 2025) is a knowledge-graph builder that induces its own schema with an LLM, a large language model. It builds a concept layer over web-scale corpora with no predefined schema. It reports 95% semantic alignment with human type inventories, measured by BS-R and BS-C, its two reported semantic-similarity scores, at 78,400 GPU-hours, hours of graphics-processor time. Its node identity is exact string match on the surface forms the model generates. It has no canonicalisation step at all, no step that merges different names for the same thing. AutoPKG, an automatic product knowledge graph builder (Hongwimol et al., 2026), is the closest prior work to our key-identity operator. It is the reason our claim is about the criterion rather than the question. It already resolves each candidate attribute key by add, merge, replace or discard over streaming listings. But it does so by a prompted LLM at temperature 0.7, a setting under which the model's output is partly random, reading a hand-written synonymy policy over ten retrieved neighbours. Its probabilistic recall collapses to 0.27 to 0.47 on the tail, the rare keys. No edit carries its evidence. Its own limitations section records that an early wrong merge propagates through every later decision. Our decisions are deterministic, and each carries a per-facet receipt in bits, one term per attribute of the record. The head-to-head under their protocol is the designed next tier (Section 7).

Per-variable decompositions, and what is new. Per-variable decompositions of information scores exist, and we do not claim them. NML, normalised maximum likelihood, is a code that pays the worst-case overhead over all datasets of a given size. Its factorised and quotient variants, fNML and qNML, are sums over variables (Silander et al., 2018). Kontkanen et al. (2003) factorise the inner term of a clustering NML into per-attribute regrets; a regret is the extra bits a code pays over the best code chosen in hindsight. But the normaliser keeps a joint sum over all attributes at once. That is one reason our spine is prequential, predicting each record before it is seen (Dawid, 1984; Grünwald, 2007), rather than NML. DIBmix, a clustering method for mixed-type data built on the deterministic information bottleneck (Costa et al., 2026), reports per-variable mutual information after the fact, on a partition whose cluster count the user fixes. Agglomerative IB (Slonim & Tishby, 1999), where IB is the information bottleneck, a method that compresses one variable

while keeping what it says about another, supplies the merge loss we reuse. It selects no partition size. Snob (Wallace & Dowe, 1994) prints per-attribute significance after fitting. What is new is not the per-facet term but its job. It is the decision variable of an online create test, computed at arrival, before the node exists. The winning term is promoted to a typed edge, an edge labelled by the attribute that decided it, carrying its cost in bits.

C FULL PROOFS

Terms used here: a prequential code prices each record by a prediction made from the past alone; Kraft's inequality says codelengths ℓ are realisable exactly when $\sum 2^{-\ell} \leq 1$; KT is the Krichevsky-Trofimov code, which grants every symbol half a count up front; the IBP is the Indian buffet process, the feature-allocation model whose creation rate γ we integrate out; a martingale is a fair game, and Ville's inequality bounds how often its wealth ever gets large; NML is normalized maximum likelihood, the minimax code that exists only when its normaliser is finite; $D(p\|q)$ is the Kullback-Leibler divergence.

This appendix carries the paper's formal content in the order of Section 4: the coding setting; the exact relation to structural entropy, concessions first; the identity that locates λ; the impossibility that forces deferral, and the price that deferral must cover; the anytime guarantee of the escrow; the identities behind the typed edges; and the batch objective that the repair pass descends. The results stated in Section 4 are restated here in full, with the number they carry there, and each proof follows its statement; the lemma on where λ comes from is stated only here, under its own number.

Three conventions hold throughout. Logarithms are base 2 unless marked, so codelengths are in bits. $H(\cdot)$ is Shannon entropy: the average number of bits an optimal code spends per symbol of a source. A proof follows every statement, and every displayed identity was also checked numerically.[1]

C.1 SETTING

A record is a set of key-value pairs. Record n publishes a key set $O_n \subseteq A$, the attributes it carries, and one value $x_{n,a} \in V_a$ for each published attribute a; the value alphabet V_a has d_a symbols. The latent structure is a feature allocation: a record may belong to several nodes at once, not to exactly one. We model the allocation with the Indian buffet process (IBP) (Griffiths & Ghahramani, 2011). In plain terms: record n attaches to an existing node k with probability m_k/n, where m_k counts the earlier records attached to k, and it opens $j \sim \mathrm{Poisson}(\gamma/n)$ new nodes, where the concentration γ sets how readily the process creates nodes. We do not fix γ. It gets a $\mathrm{Gamma}(a, b)$ prior, $a, b > 0$, and is integrated out exactly, so no node-creation rate is assumed anywhere. Values are coded by per-facet sequential Krichevsky-Trofimov (KT) predictives (Krichevsky & Trofimov, 1981): a KT predictive is a running estimate of a value distribution built from the counts seen so far, with half a count granted to every symbol up front, so no value ever has probability zero. A background node 0 carries the corpus-wide value tables, with predictive $\hat{P}_{0,a}$; it is the model of an ordinary record, and every claim of structure must beat it. Three running statistics recur. K_n is the number of nodes minted through record n. $H_n = \sum_{m \leq n} 1/m$ is the harmonic sum; under the IBP it measures accumulated exposure to mint opportunities, and it grows like $\ln n$. $\hat{\gamma}_n = (a + K_n)/(b + H_n)$ is the posterior mean concentration: the stream's own running estimate of how freely it mints. Last, the coding discipline. Prequential means that every record is priced by a predictive computed from the transmitted past alone, before the record is revealed (Dawid, 1984; Grünwald, 2007). Because

[1] The identities of Proposition C.1 have a shipped artefact, `code/experiments/e17_structural_entropy_identities.py`, which writes `results/e17_structural_entropy_identities.json`. Measured there: the closed forms of part (i), the slack identity (C.3) at tree heights one, two and three, and both forms of the gain identity (C.1) reproduce to 2.2×10^{-15} over 500 random bipartite graphs with random partitions and trees, with the two degenerate extremes equal to $H^{(1)}(B)$ to 1.8×10^{-15}; the coincidence identity (C.4) to 1.1×10^{-12} over 40 synthetic streams satisfying the hypotheses of part (ii); the Krichevsky-Trofimov regret (C.5) nonnegative in all 400 blocks drawn, at a minimum of 2.02 bits; the fibre pair of part (iv) returning an identical H^T on all 200 partitions tried, with the merge verdict *distinct* at 100.0 bits of divergence against a parameter saving of -3.32; and $\Lambda_1 = 2.000$ bits. One numerical check quoted elsewhere in this appendix still has no shipped artefact, and is marked as such where it appears: the price (C.6) against the product of prequential conditionals to 10^{-12}. The regret slope of Lemma 2(iii), 1.661 bits per decade of exposure, is quoted there as the closed form $\frac{1}{2}\log 10$ that it is, not as a measurement.

each predictive is a genuine probability distribution over what comes next, the total is a valid prefix codelength, and Kraft's inequality, the fact that probabilities summing to at most one and achievable codelengths are the same objects, governs every decision the code prices.

C.2 THE RELATION TO STRUCTURAL ENTROPY

Before any result of our own, we place the nearest incumbent objective. A reader can skip this proposition without loss: it positions a rival, and the paper's own results begin at the paragraph on where λ comes from. Structural entropy (Li & Pan, 2016) scores a hierarchical partition of a graph, called an encoding tree, by the number of bits needed to tell a random walker where it lands; it is unsupervised, entropy-shaped, free of a fixed cluster count, and minimised incrementally by HISEvent (Cao et al., 2024). The question the next proposition answers: how much of our objective does structural entropy already contain, and what can it not express? We state the overlap first, in closed form, because the two differences that survive the concession are the ones the rest of the paper stands on.

To compare the two objectives at all, the stream must become a graph structural entropy can read. An *atom* is a key-value pair $\alpha = (k, u)$: one specific value of one specific key. Derive from the stream the bipartite graph $B = (R \cup \mathcal{A}, E_B)$, records on one side, atoms on the other, with one unit edge (r, α) iff record r carries α. Degrees then have concrete meanings. A record's degree is its width m_r, the number of pairs it publishes. An atom's degree is its frequency n_α, the number of records carrying it. $M = \sum_r m_r$ is the total number of published pairs. $H^T(B)$ is the structural entropy of an encoding tree T; a height-2 tree is simply a partition $P = \{X_j\}$ of the vertices, with volume V_j (the sum of degrees inside X_j), internal weight $w(X_j)$ (the number of edges with both ends inside X_j), and $P(\alpha)$ the module containing α.

Proposition C.1 (relation to structural entropy)**.** *(i)* Closed forms, and an additivity conceded. *Write $p_R(r) = m_r/M$ for the width profile and $p_{\mathcal{A}}(\alpha) = n_\alpha/M$ for the frequency profile. Then $H^{(1)}(B) = 1 + \frac{1}{2}[H(p_R) + H(p_{\mathcal{A}})]$: the leading bit says on which side of the bipartite graph the walker lands, records or atoms, the bracket averages the two degree profiles' entropies, and the whole is a function of the two marginals alone, blind to which record carries which atom. For every partition P, with $s_{j,\alpha}$ the number of records of X_j carrying α, the gain of P over the one-level code has two equal closed forms:*

$$H^{(1)}(B) - H^P(B) \;=\; \sum_j \frac{w(X_j)}{M} \log \frac{2M}{V_j} \;=\; \sum_{\alpha \in \mathcal{A}} \frac{s_{P(\alpha),\alpha}}{M} \log \frac{2M}{V_{P(\alpha)}} \;\geq\; 0, \tag{C.1}$$

vanishing exactly at the all-singleton and one-module extremes. In words: each module X_j contributes the share of edges internal to it, $w(X_j)/M$, times the bits saved by pointing at a landing spot inside X_j rather than anywhere in the graph, $\log(2M/V_j)$; the second form books the same saving atom by atom, which is the additivity being conceded. So 2-D structural entropy is genuinely anti-degenerate, and its gain is exactly additive over attribute atoms, so it does say which attribute. We concede both facts before drawing any contrast. (ii) Coincidence. *Suppose each record activates one node v and value supports are node-disjoint; then every edge of B is internal to a module. On this overlap the two objectives become one problem: the value term of the structural entropy of the induced partition is exactly the maximum-likelihood plug-in of our value blocks. This part is an identity. The prequential blocks the engine ships stand above that plug-in by the KT code's regret per block, the extra bits a sequential code pays over the best code chosen in hindsight, and that part is an inequality: the regret is nonnegative for every block, and its size, $\frac{A_k-1}{2} \log q_{v,k} + O(1)$, is the one step of this proposition taken from the literature rather than derived here. Restricted to disjoint allocations, the value blocks, and no model cost, the two minimisations are the same problem, and every remaining term lies outside the overlap. The exact term-by-term accounting is Equation* (C.4). *(iii)* First essential difference: no model-cost term. *Every term of $H^T(B)$ is a function of cut and volume aggregates. Nothing charges for describing the tree T itself, and what blocks the all-singleton optimum is $w(X_j) = 0$, a connectivity fact, not a price. H^T is intensive, a rate; a codelength is extensive, a total. A per-arrival mint test compares bits of evidence against bits of model, and a rate supplies no such increment, which is where that line's residual stopping rules live. Before any edge exists H^T is undefined, while the price* (C.6) *is defined from record one: $\Lambda_1 = \log \frac{b+1}{a}$, two bits at the declared $(a, b) = (\frac{1}{2}, 1)$. (iv)* Second essential difference: key-type invariance. *Two datasets that differ only in how their atoms are typed into keys produce the same graph B, so no structural*

entropy at any tree height can tell them apart: H^T, its minima, and its minimisers are constant on the fibres of the map that forgets atom types, where a fibre is the set of datasets mapped to one graph. The fibres are non-trivial, and the criterion separates them: on datasets identical as graphs but typed one-key-with-two-values versus two-keys, it prices the difference at the value level in one and at the key-presence level in the other, a well-posed key-merge instance returning distinct*. The audited outputs, which key carried a difference and at which level, are computed by no function of B, at any tree height, under any edge weighting.*

The proof is elementary and every step is checkable by hand. It is given in full because the proposition concedes as much as it claims, and a reader who cannot check the concession cannot weigh the claim.

Proof. Parts (i), (iii) and (iv) rest on one telescoping identity, which is therefore established first. Let T be any encoding tree of B with root λ. For a non-root node α write $L(\alpha)$ for the graph vertices at the leaves below α, so that $V_\alpha = \sum_{i \in L(\alpha)} d_i$ and $V_\lambda = \mathrm{vol}(B) = 2M$; a leaf is a single vertex i, with $L(i) = \{i\}$ and $V_i = g_i = d_i$. Write $\pi(i)$ for the non-root nodes on the path from λ down to the leaf i. Then

$$\sum_{\alpha \neq \lambda} \frac{V_\alpha}{2M} \log \frac{V_{\alpha^-}}{V_\alpha} \;=\; \sum_i \frac{d_i}{2M} \sum_{\alpha \in \pi(i)} \log \frac{V_{\alpha^-}}{V_\alpha} \;=\; \sum_i \frac{d_i}{2M} \log \frac{2M}{d_i} \;=\; H^{(1)}(B). \tag{C.2}$$

The first equality expands V_α over the leaves below α and exchanges the two sums, which is legitimate because $\{\alpha \neq \lambda : i \in L(\alpha)\}$ is exactly $\pi(i)$. The second telescopes that path: consecutive terms cancel and only $\log V_\lambda - \log V_i = \log(2M/d_i)$ survives. The third is the definition of $H^{(1)}(B)$, the structural entropy of the unique tree of height one, whose children are the leaves themselves. Subtracting the definition $H^T(B) = \sum_{\alpha \neq \lambda} \frac{g_\alpha}{2M} \log \frac{V_{\alpha^-}}{V_\alpha}$ from (C.2) gives the slack identity

$$H^{(1)}(B) - H^T(B) \;=\; \sum_{\alpha \neq \lambda} \frac{V_\alpha - g_\alpha}{2M} \log \frac{V_{\alpha^-}}{V_\alpha} \;\geq\; 0, \tag{C.3}$$

where every summand is nonnegative because a vertex set's cut weight never exceeds its degree volume, $g_\alpha \leq V_\alpha$, and a child's leaves are a subset of its parent's, $V_\alpha \leq V_{\alpha^-}$.

Part (i), the closed forms. For the one-dimensional form, expand the definition $H^{(1)}(B) = -\sum_{x \in R \cup \mathcal{A}} \frac{d_x}{2M} \log \frac{d_x}{2M}$ over the two sides of B. A record vertex has $d_r = m_r$, hence $\frac{d_r}{2M} = \frac{1}{2} p_R(r)$, and its term is $-\frac{1}{2} p_R(r) \big[\log p_R(r) - 1\big]$; summing over R and using $\sum_r p_R(r) = 1$ gives $\frac{1}{2} H(p_R) + \frac{1}{2}$. An atom vertex has $d_\alpha = n_\alpha$ and $\sum_\alpha n_\alpha = M$, so the identical computation on the atom side gives $\frac{1}{2} H(p_\mathcal{A}) + \frac{1}{2}$. Adding the two halves gives $H^{(1)}(B) = 1 + \frac{1}{2}[H(p_R) + H(p_\mathcal{A})]$, in which the leading bit is the two halves' constants and the bracket reads the two degree marginals and nothing else.

For the two-dimensional form, let T be the height-two tree of a partition $P = \{X_1, \ldots, X_L\}$ and apply (C.3). Its leaves contribute nothing, because a leaf has $V_i - g_i = d_i - d_i = 0$. Its modules have the root as parent, so $V_{\alpha^-} = 2M$, and they satisfy $V_j - g_j = 2w(X_j)$: the volume V_j counts each edge internal to X_j twice, once at each endpoint, and each edge leaving X_j once, so $V_j = 2w(X_j) + g_j$. Substituting both facts into (C.3) gives the first equality of (C.1). For the second, use that B is bipartite: an edge internal to X_j is a pair (r, α) with exactly one atom endpoint, and that atom lies in exactly one module of P. Grouping the internal edges of X_j by their atom endpoint therefore counts each once and misses none, so $w(X_j) = \sum_{\alpha \in X_j \cap \mathcal{A}} s_{j,\alpha}$, and summing over j replaces the module index by $P(\alpha)$. Both factors of each summand are nonnegative, since $w(X_j) \geq 0$ and $V_j \leq 2M$, so the gain is nonnegative. It vanishes at the all-singleton partition, where no module holds two vertices and hence $w(X_j) = 0$ for every j, and at the one-module partition, where $V_1 = 2M$ and the logarithm is zero; it is strictly positive as soon as one module holds an internal edge and has volume below $2M$. The second form is an exact per-atom accounting of the gain, which is the additivity the statement concedes.

Part (ii), coincidence. Recall the per-node statistics: $z_{rv} = 1$ when record r activates node v, $q_{v,k}$ is the number of members of v naming key k, $Q_v = \sum_k q_{v,k}$, $c_{v,k}(u)$ is the number of members with value u on key k, $\hat{c}_{v,k}$ is the empirical value distribution inside the block (v, k), and

$\hat{\kappa}_v = (q_{v,k}/Q_v)_k$ is the node's key profile. Let P_Z have modules $X_v = \{r : z_{rv} = 1\} \cup \{\alpha : \alpha \text{ is carried by a member of } v\}$. Three steps.

First, P_Z is a partition of $R \cup \mathcal{A}$. Single membership makes the record sides disjoint. If an atom $\alpha = (k, u)$ lay in both X_v and $X_{v'}$ with $v \neq v'$, some member of each would carry it, so $c_{v,k}(u) > 0$ and $c_{v',k}(u) > 0$, contradicting the node-disjointness of value supports. Every atom is carried by some record, so the modules cover.

Second, every edge is internal and $V_v = 2Q_v$. An edge (r, α) whose record endpoint is a member of v has $\alpha \in X_v$ by construction, so both endpoints lie in X_v and $g_v = 0$ for every module. For the volume, the record side contributes $\sum_{r:z_{rv}=1} m_r = \sum_k q_{v,k} = Q_v$, because both sides count the pairs (member of v, key it names). The atom side contributes $\sum_{\alpha \in X_v \cap \mathcal{A}} n_\alpha$; disjointness makes every record carrying $\alpha = (k, u)$ a member of v, so $n_\alpha = c_{v,k}(u)$ and the sum is $\sum_k \sum_u c_{v,k}(u) = \sum_k q_{v,k} = Q_v$. Hence $V_v = 2Q_v$.

Third, the accounting. With every $g_v = 0$ the module terms of H^{P_Z} vanish and only the leaf terms remain, so $2M \cdot H^{P_Z}(B) = \sum_v \sum_{i \in X_v} d_i \log \frac{V_v}{d_i}$. The record side of that sum is $\sum_v \sum_{r:z_{rv}=1} m_r \log \frac{2Q_v}{m_r}$ directly. On the atom side, group the atoms of X_v by key and split the logarithm in three,

$$\log \frac{V_v}{c_{v,k}(u)} \;=\; \log \frac{q_{v,k}}{c_{v,k}(u)} \;+\; \log \frac{Q_v}{q_{v,k}} \;+\; \log \frac{V_v}{Q_v},$$

whose last term is $\log 2 = 1$ because $V_v = 2Q_v$. Weighting by $c_{v,k}(u)$ and summing over u and then k: the first piece gives $\sum_k q_{v,k} H(\hat{c}_{v,k})$, since $\sum_u c_{v,k}(u) = q_{v,k}$; the second gives $Q_v H(\hat{\kappa}_v)$, since summing out u leaves $\sum_k q_{v,k} \log \frac{Q_v}{q_{v,k}}$; the third gives Q_v. Summing over v and using $\sum_v Q_v = \sum_r m_r = M$ yields

$$2M \cdot H^{P_Z}(B) \;=\; \underbrace{\sum_v \sum_k q_{v,k}\, H(\hat{c}_{v,k})}_{\text{ML plug-in value cost}} \;+\; \sum_v Q_v\, H(\hat{\kappa}_v) + \sum_v \sum_{r:\, z_{rv}=1} m_r \log \frac{2Q_v}{m_r} + M, \quad \text{(C.4)}$$

an identity, with no asymptotics anywhere in its derivation.

It remains to say exactly how the first bracket relates to our value blocks, because this is the one place in the proposition where the relation is an inequality rather than an identity. Our block for the pair (v, k) is the sequential KT code over the A_k symbols of the key, whose codelength on the counts $c_{v,k}$ is exactly

$$L_{\mathrm{KT}}(c_{v,k}) \;=\; \log \frac{\Gamma\big(q_{v,k} + A_k/2\big)\, \Gamma(1/2)^{A_k}}{\Gamma(A_k/2)\, \prod_u \Gamma\big(c_{v,k}(u) + 1/2\big)},$$

which is an identity, and which is what `codes.L_KT_block` computes. The first bracket of (C.4) is instead the maximum-likelihood plug-in $q_{v,k} H(\hat{c}_{v,k})$, which is not a codelength at all: it prices the counts by their own empirical frequencies, chosen after the block is read. The two are related in one direction unconditionally. The KT code is a mixture $\int P_\theta(c_{v,k})\, w(\theta)\, d\theta$ over the value distributions θ with w the Dirichlet$(\frac{1}{2}, \ldots, \frac{1}{2})$ density, and a mixture cannot exceed its largest member, so $P_{\mathrm{KT}}(c_{v,k}) \leq \max_\theta P_\theta(c_{v,k}) = 2^{-q_{v,k} H(\hat{c}_{v,k})}$ and therefore

$$L_{\mathrm{KT}}(c_{v,k}) \;-\; q_{v,k}\, H(\hat{c}_{v,k}) \;\geq\; 0 \quad \text{(C.5)}$$

for every block and every count vector. That this excess is $\frac{A_k - 1}{2} \log q_{v,k} + O(1)$ and not merely nonnegative is the standard KT result, which we cite (Krichevsky & Trofimov, 1981; Xie & Barron, 2000) rather than reprove; it is the only step of this proposition taken from the literature. The claim of part (ii) is therefore exactly this strong and no stronger. The first bracket of (C.4) is the plug-in of our value blocks term for term, so restricted to a disjoint allocation, to the value blocks alone and with the model cost dropped, the two minimisations are one problem; the blocks the engine ships sit above that bracket by (C.5); and every other term of (C.4) lies outside the overlap. Structural entropy adds the key-landing term $\sum_v Q_v H(\hat{\kappa}_v)$ and the record-landing term, which price the codelength of a random walk's landing key and landing record rather than of the records themselves, and whose nearest relatives in our code, the per-(v, k) presence blocks and the record-width code, price different

events. Our criterion adds the model cost that structural entropy has nowhere to put: the price (C.6) and the parameter half-logarithms.

Part (iii), no model-cost term. That no term of $H^T(B)$ charges for describing T is read off the definition: each summand is $-\frac{g_\alpha}{\mathrm{vol}(B)} \log \frac{V_\alpha}{V_{\alpha^-}}$, a function of $(g_\alpha, V_\alpha, V_{\alpha^-}, \mathrm{vol}(B))$ alone, and none of those four is a description of the tree. What blocks the all-singleton optimum is not a price for the parts but $w(X_j) = 0$ in (C.1), a connectivity fact about an edge set that is handed to the objective. In the gain form the quantity standing where a price would stand is the module's address length $\log \frac{2M}{V_j}$, saved once per unit of internal incidence: it decreases in the module's degree volume, is identical for every atom in the module, and does not depend on how much data supports the module. The price (C.6) is by contrast a function of the arrival clock and of two global counters of the stream, and no degree volume enters it.

The statement about units follows from (C.3). Every summand there is nonnegative, so $H^T(B) \leq H^{(1)}(B)$ for every tree T at every height; every summand of H^T itself is nonnegative for the same two reasons, so $H^T(B) \geq 0$; and $H^{(1)}(B)$ is the Shannon entropy of the degree distribution $x \mapsto d_x/2M$ on the $|R \cup \mathcal{A}|$ vertices, so $H^{(1)}(B) \leq \log |R \cup \mathcal{A}|$. Together, $0 \leq H^T(B) \leq H^{(1)}(B) \leq \log |R \cup \mathcal{A}|$. Structural entropy is thus intensive, a rate capped by the logarithm of the vertex count, while a two-part codelength is extensive and grows with the stream. A per-arrival mint test compares bits of evidence against bits of model, and a rate supplies no such increment, which is where the residual stopping rule of that line lives: HISEvent searches its one-dimensional curve for a first stable point (Cao et al., 2024). Two consequences close the part. The anti-degeneracy devices differ in kind, structural entropy's being the cut of an edge set that already exists and ours being code validity, the price plus predictive smoothing. And before any edge exists $\mathrm{vol}(B) = 0$, so every term of H^T divides by zero and no structural-entropy quantity is defined, let alone able to price the first node, whereas (C.6) is defined at $s = 1$, where $K_0 = 0$ and $H_0 = 0$ give $\Lambda_1 = \log \frac{b+1}{a}$, which is 2 bits exactly at the declared $(a, b) = (\frac{1}{2}, 1)$.

Part (iv), key-type invariance. Let Φ send a typed dataset D to the graph B with its atoms as unlabelled vertices. Every term of H^T is a function of cut and volume aggregates of B, and both are determined by $\Phi(D)$; so for every tree T, every height d and every weighting of the edges of B, the value H^T, the minimum $H^{(d)}$ and the set of minimising trees are constant on each fibre of Φ.

It remains to exhibit a fibre on which our criterion is not constant, which also shows the fibres are non-trivial. Fix $t \geq 1$ and take $2t$ records, t of them carrying the single atom α and t carrying the single atom β. Dataset D types the two as one key with two values, $\alpha = (k, u)$ and $\beta = (k, u')$; dataset D' types them as two keys with one value each, $\alpha = (k_1, u)$ and $\beta = (k_2, u')$. Both give the same graph: $2t$ record vertices of degree one, two atom vertices of degree t, and $M = 2t$. So $\Phi(D) = \Phi(D')$ and every structural-entropy computation returns the same output on the pair. Our criterion does not. In D the two groups differ inside one block: a record of the second group scored against a node holding the first pays the value-level mismatch $\log \frac{1/2}{q_{v,k}+A_k/2} = -\log(2q_{v,k}) + O(1)$ inside the block (v, k), and no key-merge question exists, there being one key. In D' the node holding the first group has no block for k_2 at all, so the same difference is priced by the key-presence term, and the pair (k_1, k_2) is a well-posed instance of the key-merge test. That test weighs the divergence a merge must pay against the parameters a merge saves. The two value distributions are point masses on disjoint supports, so for any mixing weight π,

$$\mathrm{JS}_\pi(\hat{p}_{k_1}, \hat{p}_{k_2}) \;=\; H\big(\pi\delta_u + (1-\pi)\delta_{u'}\big) - \pi H(\delta_u) - (1-\pi)H(\delta_{u'}) \;=\; H(\pi),$$

a point mass having zero entropy (Lin, 1991). At $q_{k_1} = q_{k_2} = t$ the weight is $\pi = \frac{1}{2}$, so $H(\pi) = 1$ and the divergence side is $Q\,\mathrm{JS}_\pi = 2t$ bits. The parameter side is $\frac{|V_{k_1}|-1}{2} \log q_{k_1} + \frac{|V_{k_2}|-1}{2} \log q_{k_2} - \frac{|V_{k_1} \cup V_{k_2}|-1}{2} \log Q = 0 + 0 - \frac{1}{2} \log 2t$, which is negative for every $t \geq 1$. The test therefore returns *distinct*, at every t. The two audited outputs, which key carried the difference and whether it was priced at the value level or at the presence level, take different values at the two points of one Φ-fibre; hence no function of B, at any tree height and under any edge weighting, computes them. □

The consequence is narrow, and it is not construction versus partition: HISEvent constructs its own edge set. A procedure whose output is an encoding tree of its input graph cannot output the escrowed allocation, for three reasons now visible. Nothing in its run prices the birth of a new vertex. Its trees

partition where the allocation covers. And by (iv) it can emit no typed edge or key-merge verdict. HISEvent is scheduled as a baseline in the designed external tier (Appendix G.18).

C.3 WHERE λ COMES FROM

The question of this subsection: where does the field's free parameter come from? The answer is an identity, and it says that λ is not a loose end of the BP-means derivation. It is the only part of the new-node price the derivation keeps. BP-means is the small-variance limit of exactly the model above (Broderick et al., 2013b). Small-variance asymptotics means: put a Gaussian emission model with noise variance σ^2 on the features, multiply the negative log joint by $2\sigma^2$, and let $\sigma \to 0$; what survives is a k-means-style objective in which opening a new feature costs a penalty λ^2. The recipe is called MAD-Bayes in Broderick et al. (2013b), and we use that name below. That penalty is the object we replace. In the lemma, K_+ is the number of new features a step opens, D the feature dimension, ρ the prior width on a feature's parameters, N the number of records, and S_k the number of records attached to feature k. The word mass below is IBP vocabulary for the concentration parameter γ, the rate that sets how readily new features open, and the scaling of γ with σ in the lemma is the one Broderick et al. (2013b) themselves require for the limit to keep a finite nonzero penalty: it is theirs, not a choice of ours.

Lemma C.1 (the limit deletes the computable price)**.** *The BP-means mass scaling* $\gamma = \exp(-\lambda^2/(2\sigma^2))$ *makes the following identity exact:*

$$\lambda^2 \;=\; 2\sigma^2 \cdot (-\ln \gamma) \;=\; 2\sigma^2 \times (\textit{prior codelength, in nats, of one new node}).$$

So λ^2 *is a codelength in disguise, in units of* $2\sigma^2$*. The limit then deletes every computable part of that price. The negative log joint carries three new-node terms: the mass term* $-K_+ \ln \gamma$*, engineered by the scaling to be* $O(1/\sigma^2)$*; the parameter cost* $(K_+ D/2)\ln(2\pi\rho^2)$*; and the allocation cost* $\sum_k \ln \binom{N}{S_k}$*, the two* $O(1)$ *terms of the MAD-Bayes bookkeeping. Multiplying by* $2\sigma^2$ *and letting* $\sigma \to 0$ *annihilates exactly the two computable parts and keeps the part inserted by hand. The Chinese restaurant process (CRP), the single-membership analogue of the IBP from which DP-means descends, corroborates: DP-means (Kulis & Jordan, 2012) requires* $\alpha = (1+\rho/\sigma)^{D/2}\exp(-\lambda/(2\sigma))$*, in the cited paper's own notation, where* σ *denotes the noise variance and the penalty is* λ*, so both exponents here are the penalty over twice the noise variance and the apparent asymmetry with the BP-means scaling is notational. And* $(D/2)\ln(1+\rho/\sigma)$ *is the parameter codelength of a fresh* D*-dimensional mean at the resolution the data supports, so* α *factorises as (parameter codelength)* $\times$ *(free constant), with* λ *the residue after the codelength is divided out.*

Proof. The scaling makes the mass term $O(1/\sigma^2)$ by construction, so $2\sigma^2$ times it tends to $K_+\lambda^2$; the parameter and allocation costs are $O(1)$ in σ, so $2\sigma^2$ times them tends to zero. The DP-means factorisation is read off the cited requirement on α by taking logarithms. □

In words: the exact model contains a principled, computable price for a new node, the parameter bits plus the allocation bits, and the limit that produces BP-means subtracts precisely those two terms. What survives is the one term the scaling was engineered to protect. λ is a free parameter because the limit was taken, not despite it. Our answer is to not take the limit: code the same model prequentially and read the price off the code. Two results say what that forces.

The first result answers the question a reader should ask at once: why not simply create a node the first time a record fails to fit? Because a brand-new node has nothing to predict with. Under any honest code its first prediction is uninformed, so its first record cannot, in expectation, out-code a background that has seen the whole corpus. The one apparent escape, seeding the node's predictive at the observed value itself, only relocates the same bits into the seed. One more term is needed to state this: $D(p \,\|\, q)$ is the Kullback-Leibler divergence, the expected number of extra bits paid for coding data drawn from p with a code built for q; it is zero iff $p = q$ and positive otherwise. And one property: a prior is exchangeable over values, or value-exchangeable, when it treats the alphabet's symbols interchangeably, favouring none before data arrive.

Lemma 1 (myopic impossibility), restated in full. Let Q be the predictive a node minted at record n assigns to a value of attribute a before any record has attached to it. (i) In a valid code

Q is measurable with respect to the transmitted past, with $\sum_x Q(x) \leq 1$. With no past support, exchangeability over values forces Q uniform, so every value costs $\log d_a$; the KT table at zero counts realises this. A construction that seeds Q at the observed value is a two-part code whose seed itself costs at least $\log d_a$ under any value-exchangeable prior: the same bits, relocated. (ii) With p the conditional law of $x_{n,a}$, the expected first-record evidence against the background is $-D(p\|Q) + D(p\|\hat{P}_{0,a})$. Under the operational null $p = \hat{P}_{0,a}$ this equals $-D(\hat{P}_{0,a}\|Q) \leq 0$; for $Q = \mathrm{Uniform}(d_a)$ it is $-D(\hat{P}_{0,a}\|\mathrm{Unif})$, zero iff the background is uniform. (iii) $\Lambda_n > 0$ whenever $a + K_{n-1} < n(b + H_{n-1}) + 1$. So the expected net value of minting on the arriving record is at most $-\Lambda_n < 0$ on every attribute: under any valid code a fresh node cannot pay for itself, in expectation, on its own first record. The only pointwise exceptions are rarity events, values the background itself prices below $2^{-\Lambda_n}/d_a$, whose codelength signature is identical for a novel entity and a corrupted record.

The lemma is not only a warning on paper: E7 (Section 5.2) measures both immediate forms at zero mints on 6,000 records; three earlier derivation routes that reported $K \in \{0, 1\}$ are development notes without a shipped artefact and are not counted as evidence here. DP-means has no such obstacle only because it places the new centroid at the point, so the new cluster's data cost is zero and one distant record already pays λ; a predictive code has no such move. The repair is the deferred mint of Section 3.

Lemma 1 states that under any valid code a fresh node cannot pay for itself, in expectation, on its own first record. The proof establishes its three parts: (i) a fresh node's predictive is forced uniform, (ii) its expected first-record evidence is nonpositive, and (iii) the price it must beat is strictly positive.

Proof. (i) is Kraft's inequality plus exchangeability. A valid predictive Q for a fresh node is measurable with respect to the transmitted past and satisfies $\sum_x Q(x) \leq 1$. If the past carries no support for any value of attribute a, exchangeability over values forces Q uniform, so every value costs $\log d_a$. The Krichevsky-Trofimov (KT) code, the standard Bayes code that prices each value by its smoothed running count, starting every count at one half, realises this exactly at zero counts (Krichevsky & Trofimov, 1981). Any construction that instead seeds Q at the observed value is a two-part code whose seed must itself be transmitted, at a cost of at least $\log d_a$ under any value-exchangeable prior: the same bits, relocated.

(ii) asks what the fresh node is expected to save against the background predictive $\hat{P}_{0,a}$, and the answer is the identity $\mathbb{E}_{x\sim p}[\log Q(x) - \log \hat{P}_{0,a}(x)] = -D(p \,\|\, Q) + D(p \,\|\, \hat{P}_{0,a})$. Evaluated at the operational null $p = \hat{P}_{0,a}$, it equals $-D(\hat{P}_{0,a} \,\|\, Q) \leq 0$, with equality only when the evidence is identically zero. For the minimax choice $Q = \mathrm{Uniform}(d_a)$ it is $-D(\hat{P}_{0,a} \,\|\, \mathrm{Unif}) \leq 0$, with equality iff the background is uniform. In words: on its first record the fresh node is expected to lose bits, never gain them.

(iii) is immediate from Lemma 2: $2^{-\Lambda_s} = (a + K_{s-1})/(s(b + H_{s-1}) + 1)$, which is below one whenever $a + K_{s-1} < s(b + H_{s-1}) + 1$; at $s = 1$ and the declared $(a, b) = (\frac{1}{2}, 1)$, $\Lambda_1 = \log \frac{b+1}{a} = 2$ bits exactly.

Combining the three parts, the mint route is dominated in expectation by the all-background action by at least Λ_n bits on every attribute. Pointwise, a seeded mint saves bits only on values with $\hat{P}_{0,a}(x) < 2^{-\Lambda_n}/d_a$. That event is a statement about rarity alone, and its codelength signature is identical for a novel entity and a corrupted record. Empirically, E7 records zero mints for both immediate forms (Section 5.2). □

The second result is the price itself. The question it answers: if the model is kept exact, what does one new node cost? The price is the log posterior odds of no mint against one mint, in bits: how much evidence a candidate must produce before the model prefers that it exist. It has a closed form, and the form is the claim. The coefficient on $\log s$ is exactly one, forced by the model, not chosen; everything else is a statistic the stream computes about itself. One coding term appears in part (i): normalized maximum likelihood (NML), the minimax-optimal code for a model class, which exists only when its normalizer, a sum of maximised likelihoods over all possible data, is finite.

Lemma 2 (the price), restated in full. Under the IBP with $\gamma \sim \mathrm{Gamma}(a, b)$ marginalised, the price of one new node at record s, the log posterior odds of no mint against one mint, is Bayes-exact:

$$\Lambda_s \;=\; \log\big(s(b + H_{s-1}) + 1\big) - \log(a + K_{s-1}) \;=\; \log s - \log \hat{\gamma}_{s-1} + O\big(1/(s \log s)\big), \quad \text{(C.6)}$$

so the constant on $\log s$ is exactly one, and the remainder is minus the log of the posterior mean concentration, a statistic of the stream itself: streams that mint often lower future prices, sparse streams raise them. Attachment odds contain no γ, so marginalisation leaves every attachment untouched: the mint-time price is purely structural. Moreover: (i) Propriety of the hyperprior is forced by Kraft. The NML normalizer over the node count sums, for every possible count K, the likelihood of K under the rate chosen best in hindsight; for a Poisson count that maximised likelihood is $K^K e^{-K}/K!$, so the normalizer is $\sum_K K^K e^{-K}/K!$, which diverges, so no NML code for the allocation exists at any n (Shtarkov, 1987), and every improper hyperprior fails to be a code: $\pi(\gamma) \propto 1/\gamma$ has infinite mass, and at $b = 0$ the first-record predictive puts probability zero on every finite mint count. (ii) The hyperprior influence is $O(1)$. The per-mint difference between proper hyperpriors is $O(1/\log n)$, 0.69 bits at $n = 10$ falling to 0.19 at $n = 10^5$ between $(\frac{1}{2}, 1)$ and $(2, 0.1)$ along the trajectory $K_s = 2H_s$. The shape $a = \frac{1}{2}$ is Jeffreys-canonical (Xie & Barron, 2000), and the scale $b > 0$ is the one residual freedom, forced positive by Kraft, declared rather than calibrated: the code is a prior (Wallace & Patrick, 1993). (iii) Learning the concentration inside the code costs $\frac{1}{2}\log H_N + O(1) = \frac{1}{2}\log\ln N + O(1)$ bits over the entire stream, negligible against a single mint's price.

The proof uses the IBP exchangeable partition probability function of Broderick et al. (2013a), the closed form for an allocation's probability that depends only on its counts, and the least-favorable property of the Jeffreys prior, the fact that it is the prior a worst-case opponent would pick, which is what makes it the minimax default (Clarke & Barron, 1994).

Lemma 2 states the closed form of the mint price and three facts about the hyperprior: (i) it must be proper, (ii) its influence is $O(1/\log n)$ per mint, and (iii) learning the concentration inside the code costs $\frac{1}{2}\log H_N + O(1)$ bits over the whole stream. The proof derives the closed form first, then proves (i), (ii) and (iii) in turn.

Proof. We first identify where γ lives in the model. The exchangeable feature probability function of the Indian buffet process is the probability of an allocation pattern; at unit mass it reads $P(Z_{1:N} \mid \gamma) = W(Z_{1:N})\,\gamma^{K_N} e^{-\gamma H_N}$ (Griffiths & Ghahramani, 2011; Broderick et al., 2013b), with W collecting every factor free of γ. So γ sits in a one-dimensional exponential family: the data touch it only through the node count K_N and the exposure H_N. With a proper $\mathrm{Gamma}(a, b)$ prior the marginal is $m(Z_{1:N}) = W(Z_{1:N})\,\frac{b^a}{\Gamma(a)}\,\frac{\Gamma(K_N + a)}{(H_N + b)^{K_N + a}}$. This is a probability distribution, so $-\log m$ is a codelength realizable by arithmetic coding. By Poisson-Gamma conjugacy the posterior before record s is $\gamma \mid Z_{1:s-1} \sim \mathrm{Gamma}(a + K_{s-1},\, b + H_{s-1})$, and the number of new nodes at record s is negative binomial under that posterior. The question the price answers is: what are the posterior odds of one new node against none at record s? They are

$$\frac{P(j{=}1)}{P(j{=}0)} \;=\; \frac{a + K_{s-1}}{s(b + H_{s-1}) + 1}.$$

The price is the negative base-2 logarithm of this ratio, which is the displayed closed form (verified to machine precision against the product of prequential conditionals, a check made during development for which no artefact ships). The asymptotic expansion in the lemma follows from $\log(s\beta + 1) = \log s + \log\beta + \log(1 + 1/(s\beta))$ with $\beta = b + H_{s-1} = \Theta(\log s)$. Finally, the attachment odds $m_k/(n - m_k)$ contain no γ, so marginalising γ leaves every attachment decision untouched: the mint-time price is purely structural.

For (i), propriety is forced. Kraft's inequality is the fact that valid codeword lengths must satisfy $\sum 2^{-\mathrm{length}} \le 1$; equivalently, only a (sub)probability can be a code. Two alternatives to the proper prior fail it. First, the normalized maximum-likelihood route: summing W over the allocations with K columns gives $\sum_{Z:K} W(Z) = H_N^K/K!$, because the sum must reproduce the $\mathrm{Poisson}(\gamma H_N)$ marginal over the node count. The Shtarkov sum, the normaliser such a code would need (Shtarkov, 1987), is therefore $\sum_K \sup_\gamma \gamma^K e^{-\gamma H_N} H_N^K/K! = \sum_K K^K e^{-K}/K!$. By Stirling its terms fall only like $1/\sqrt{2\pi K}$, so the partial sums grow like $\sqrt{2K/\pi}$ (verified numerically) and the sum diverges: no normalized maximum-likelihood code for the allocation exists at any n. Second, the

improper-prior route: for improper π, $\sum_Z \int P_\gamma \, \pi = \int \pi = \infty$, so the mixture is not a code; and at $b = 0$ the first-record predictive has $\mathbb{E}[K_1^+] = a/b \to \infty$ and puts mass 0 on every finite count. The same Kraft logic that kills batch profiling of γ forces the mixture and forces it proper.

For (ii), the shape and the sensitivity. The Fisher information is $I(\gamma) = H/\gamma$, so Jeffreys' prior is $\propto \gamma^{-1/2}$, parameterisation-invariant and asymptotically least favourable (Clarke & Barron, 1994). How much can the choice among proper hyperpriors matter? The per-mint difference between two of them is $\log \frac{b+H}{b'+H} - \log \frac{a+K}{a'+K} \to 0$ at rate $O(1/\log n)$. Between $(a, b) = (\frac{1}{2}, 1)$ and $(2, 0.1)$, evaluated through the shipped price (`results/e15_theory_numbers.json`), the exact difference is 0.69 bits per mint at $n = 10$ and 0.19 at $n = 10^5$ along the trajectory $K_s = 2H_s$, and 0.91 and 0.27 along $K_s = H_s$; the displayed asymptotic difference gives 0.68 and 0.90 at $n = 10$ and the same values at $n = 10^5$, so the form must be stated as well as the trajectory. Neither trajectory is one our runs walk: on the planted stream the engine holds no node at ten records and estimates a rate of 0.66 at a hundred thousand, where the same difference is 2.36 and 0.34 bits. The total codelength difference over the stream converges to the log ratio of the priors at the posterior mean, an $O(1)$ quantity.

For (iii), the cost of learning γ inside the code. The mixture's regret, the extra bits it spends against the best fixed γ chosen in hindsight, is $\frac{1}{2} \log H_N + O(1)$, because γ's effective sample size is the exposure $H_N \approx \ln N$. The regret slope is the closed form $\frac{1}{2} \log 10 = 1.661$ bits per decade of exposure. □

Two-part integer-code implementations replace the exact price by $c \log n$, with $c \in [1, 2]$ fixed by whichever integer code the implementation adopts (Rissanen, 1983). The Bayes-exact constant is $c = 1$. E12 measures what the choice of c, and of the other code families beside it, actually costs: on planted structure the minted set does not move at all, nine valid codes returning the same eight nodes at adjusted Rand index 1.0 with birth times shifting by up to 217 records, while on a shared-key stream and on the Wikipedia infoboxes the partition moves to a worst pairwise index of 0.6055 and 0.5233, against 0.6068 and 0.5804 over twenty arrival orders of the shipped code. The claim that the code choice changes when a node is born rather than which nodes exist therefore holds where the structure is identifiable, and not beyond it (Appendix D).

The impossibility and the price together force the mechanism: deferral. A candidate node is named by an attribute and value, its price is frozen at seed time, and evidence accrues in escrow, a ledger held for the candidate, until it covers the price. The theorem below answers the question every deferred test must face: if a candidate can be re-tested after every arrival, forever, why does the chance of a false mint not grow with the stream? Three ideas carry the answer, and each has a plain form. First, the evidence G_t is an accumulated log-likelihood ratio: the bits saved so far by coding the stream with the candidate rather than without it. Second, under the null hypothesis that the candidate is spurious, 2^{G_t} is a nonnegative martingale started at one: a process whose expected next value equals its current value, the formal version of a fair game, and a gambler playing a fair game does not get rich. Third, Ville's inequality is the anytime form of Markov's inequality: a nonnegative martingale started at one ever exceeds 2^{Λ} with probability at most $2^{-\Lambda}$, where ever ranges over the whole infinite future. The bound covers the running supremum, so re-testing at every arrival costs nothing extra. What remains is the size of the hypothesis space, and that is what the naming of candidates buys: attribute-value pairs form a vocabulary that grows with the schema, never with the record count, so the lifetime bound is independent of stream length.

Theorem 1 (anytime validity of the escrow), restated in full. Fix a candidate seed $p = (a^\dagger, v^\dagger)$ with seed time s and frozen price Λ_s computed from (K_{s-1}, H_{s-1}). Let $G_t(p)$ be the accumulated log-likelihood ratio, from s onward, between two prequential distributions on the stream: the graph with p (its accumulated posterior predictives, with the attachment and absence terms of its allocation code) against the current graph without p. Then: (i) If both are exactly normalised sequential probability assignments, then under the null $H_0(p)$, that the without-p model generates the stream, $M_t = 2^{G_t(p)}$ is a nonnegative martingale with $M_s = 1$, a test martingale in the sense of Ramdas et al. (2023). The condition is about the value block's third stage. A block holding u distinct values in N observations under a key whose inventory holds inv values retains mass $1 - u(u + \frac{1}{2})/((N + 1)(\mathrm{inv} + 1))$ under the shipped charge of $\log(\mathrm{inv} + 1)$ bits for the escape, because only $\mathrm{inv} - u + 1$ slots are reachable; the presence and absence terms are an exactly nor-

malised KT Bernoulli, so all of the deficiency lives in the value blocks. Charging $\log(\mathrm{inv} - u + 1)$ instead returns every block to mass one and makes the step exact: the one-step expectation of the ratio under the incumbent is 1.000000 over $955{,}200$ null key-steps, against 0.862862 for the shipped charge (E16, `results/e16_supermartingale_check.json`; the closed form is verified against the shipped code by brute-force summation over the alphabet on 400 random blocks, maximum error 4.4×10^{-16}). The shipped charge is a valid prefix code either way: its codeword lengths satisfy Kraft with inequality rather than equality, which wastes a little code space and is ordinary. The tight charge is implemented behind an environment flag and is not the default, because it buys the per-step identity and nothing else that matters: it preserves every positive control and improves Wikipedia purity from 0.9665 to 0.986, but moves the Wikipedia background from 2 records of 320 to 114, and it does not rescue (ii). (ii) In the idealised setting of (i), and on the whole stream rather than a chosen subsequence, Ville's inequality (Ville, 1939) gives a lifetime guarantee with no union bound over time:

$$\mathbb{P}\big(\exists\, t \geq s : G_t(p) \geq \Lambda_s\big) \;\leq\; 2^{-\Lambda_s} \;=\; \frac{a + K_{s-1}}{s(b + H_{s-1}) + 1} \;<\; \frac{\hat{\gamma}_{s-1}}{s}.$$

So a candidate may be re-tested after every arrival, forever, while spending the false-mint budget of one test. This part is stated for the idealised setting and is *not* claimed for the engine, for the reason given after the proof: the engine accrues only on records carrying the seed pattern, so its released statistic is computed on a subsequence the data chose. The measured exceedance is reported in its place. If at most one fresh seed were opened per record, the expected number of spurious mints over the whole stream would be at most $\sum_{s \leq N} \hat{\gamma}_{s-1}/s \approx \hat{\gamma} H_N = \mathbb{E}[K_N \mid \gamma = \hat{\gamma}]$, the node count the model class already expected; the shipped engine opens one candidate per residual (key, value) pair, so the count is governed by (iii). The shipped gate releases at the price of Equation D.10 alone, without the seed-time term Λ_s, and the same inequality applies at the level $c = \min_t \mathrm{Price}(t, T, n)$. (iii) (Stream-length independence.) The hypothesis space is indexed by seed pairs $(a^\dagger, v^\dagger)$, one candidate per residual pair: a set that grows with the schema and, under the open-alphabet escape, with the value inventory, never with the record count as such. In the shipped price the naming charge is the support term $L_{\mathrm{supp}}(T)$ and the node name $\log(K + 1)$ of Equation D.10 rather than a separate $\log|A| + \log d_{a^\dagger}$; either is Kraft-tight, and by Kraft the charge is the multiple-comparisons correction: added to each candidate's price, it makes the family-wise budget no larger than the worst single-candidate budget, so at the declared hyperprior the expected number of spurious mints, over every candidate and the whole lifetime, is at most $\max_p 2^{-\Lambda_p} < 1$: less than one spurious node in expectation, ever. A deployment may also raise every candidate's price by the same extra Γ bits, a uniform per-candidate slack on top of the naming charge, to buy a smaller bound; the shipped system adds none. With that slack, $\mathbb{P}(\text{any spurious mint, ever}) \leq |A|\, \bar{d}\, 2^{-\Gamma}$, independent of the stream length T. Under mint-on-first-miss, by contrast, the hypotheses are one per record, a set of size T: naming costs $\Theta(\log T)$ and the false-mint bound grows as $\Theta(T)$ unless the charge is allowed to know the stream length, a tuned parameter. The reparameterisation from records to seed pairs exists only under the IBP stance: a feature is named by an attribute and value, whereas a DP-means cluster is named by a datapoint.

Theorem 1 states that the escrow can re-test a candidate after every arrival, forever, while its lifetime probability of minting a spurious node stays below $2^{-\Lambda_s}$, and that the bound over all candidates together does not grow with the stream. The proof shows the evidence process is a martingale, applies Ville's inequality, and then pays for the union bound in bits.

Proof. (i) Write $M_t = \prod_{u=s}^{t} Q_u(x_u)/P_u(x_u)$, where Q_u and P_u are the one-step predictives of the graph-with-p and without-p models. Each is normalised given $\mathcal{F}_{u-1}$, the information available before record u, because every factor of a prequential code is a probability distribution computable from the transmitted past (Dawid, 1984), under the tight naming charge of the statement. Under the shipped charge both are sub-probabilities, and the exact one-step expectation under the incumbent read as a normalised law is the ratio of the two masses, Z_Q/Z_P, so the accumulator is a supermartingale exactly when $\phi(u_Q, N_Q) \geq \phi(u_P, N_P)$ on every published key with $\phi(u, N) = u(u + \frac{1}{2})/(N + 1)$: a condition whose direction is not fixed, holding at 99.5 percent of steps on a low-cardinality null and failing at 99.7 percent on Wikipedia infoboxes. A martingale is a process whose expected next value, given everything so far, equals its current value. Under the null

$H_0(p)$ the next record is drawn from P_t, so

$$\mathbb{E}[M_t \mid \mathcal{F}_{t-1}] \;=\; M_{t-1} \sum_x P_t(x)\, \frac{Q_t(x)}{P_t(x)} \;=\; M_{t-1} \sum_x Q_t(x) \;=\; M_{t-1}.$$

The ratio cancels the true law exactly, so M_t is a nonnegative martingale with $M_s = 1$.

(ii) Ville's inequality (Ville, 1939) says that a nonnegative martingale starting at one ever exceeds a level c with probability at most $1/c$, over its whole infinite future. Applied at $c = 2^{\Lambda_s}$ it gives $\mathbb{P}(\sup_t M_t \geq 2^{\Lambda_s}) \leq 2^{-\Lambda_s}$, with no union bound over time; this is the standard anytime-valid construction (Ramdas et al., 2023). The closed form and the bound $\hat{\gamma}_{s-1}/s$ follow from Lemma 2. Summing $\hat{\gamma}_{s-1}/s$ over seed times gives approximately $\hat{\gamma} H_N$ when $\hat{\gamma}$ is stable, while $\mathbb{E}[K_N \mid \gamma] = \sum_{n \leq N} \gamma/n = \gamma H_N$ under the prior itself. In words: the escrow's lifetime false-mint budget equals what the model class already expected to spend on nodes.

(iii) is the union bound, paid for in bits. If candidate p must clear $\Lambda_p + \log|A| + \log d_{a^\dagger}$, then $\sum_p 2^{-\Lambda_p - \log|A| - \log d_{a^\dagger}} \leq \max_p 2^{-\Lambda_p}$, since there are $d_{a^\dagger}$ candidates per key. Equivalently, with a uniform per-candidate slack Γ the family bound is $|A|\,\bar{d}\,2^{-\Gamma}$. The counts of the two hypothesis spaces are as stated: seed pairs number $|A|\bar{d}$ and grow only as new values appear, while records number T. Only the seed-pair indexing is available without labels: it is the Indian buffet process's feature naming, not an algorithmic convenience. □

One caveat on the implemented statistic. Validity requires the escrow to be computed on the full stream, never on a post-selection subsequence: conditioning on per-record attachment counts breaks the complete randomness the martingale rests on, the property of the IBP that lets disjoint parts of the stream be priced independently (see Doshi-Velez & Williamson (2015) for why restriction breaks it). The implementation accrues only on records carrying the pattern, dropping the absence terms $\log(1 - m_p/u)$, where m_p is the candidate's attachment count and u the arrival index at which the term would be charged; under the IBP the record at index u fails to attach to the candidate with probability $1 - m_p/u$, so each dropped term is the log-probability of one non-attachment. Dropping negative terms can only inflate $G_t(p)$, by at most $O(m_p \log(t/s))$ bits, an anti-conservative omission on its own. The guarantee holds for the implemented statistic because exactly that bound is charged back on the price side, through the cohort-membership bits $\log\binom{n}{t}$ of the column code. The accounting, in one line: the omission can add at most $O(m_p \log(t/s))$ bits to the evidence side, and the gate the implementation enforces is the price of Equation D.10 alone (Equation 2), without the seed-time term Λ_s, whose statement terms carry the column code's $\log\binom{n}{t}$ bits on the price side; the inflation is dominated by that charge, so the release fires no earlier than an honest accounting would allow. One gap does not close in this draft, and it is the reason part (ii) is stated for the idealised setting only: the released statistic is the positive-part prefix over keys sorted by escrow plus the outside absence credit (Section 3), accrued only on records carrying the seed pattern, and no separate term pays for selecting either the positive subset or that subsequence. The argument for the omitted absence terms is the justification of the implemented statistic in Appendix C.

The exceedance, measured rather than claimed. Because the bound does not transfer, the paper reports what the engine does. Over 300 independent null streams of 2,000 records and 12,000 candidate trajectories, $\Pr(\sup_t G_t \geq b)$ at $b = 1, 2, 4, 8, 12$ is 1.000 at every level for the shipped statistic; 1.000 at every level with the tight naming charge alone; 0.727, 0.583, 0.316, 0.0498, 0.0032 with the seed key's own term excluded alone; and 0.400, 0.309, 0.170, 0.0395, 0.0077 with both, against the nominal 0.5, 0.25, 0.0625, 0.0039, 0.00024 (standard error about 0.004). Both corrections together hold only at $b = 1$. The dominant cause is therefore selection, not the code: the seed key's value is constant on every record the engine accrues on, so its term is positive by construction and is not a fair bet.

The practical no-false-mint result does not depend on any of this, and it is stronger than the bound would be. The release gate fires on the computed price of Equation D.10, whose membership column grows faster than the accumulated drift, and no node is minted on any of the 48 null streams of E8, nor on any of the 300 wider streams, under all four combinations of the two flags.

Excluding the seed key's evidence, which is the only one of the two corrections that moves the exceedance, is not available as a default, and the reason is a finding rather than an inconvenience.

On the planted eight-group stream at its shipped noise rate the unselected statistic returns no nodes at all, and over twenty arrival orders the node count averages 1.6 against a planted eight; the seed key's term is about 43 percent of a candidate's evidence, 332 bits of 777 in the diagnostic. The engine's drift is load-bearing: the rule mints because the pattern that created the candidate keeps paying, and the price, not the statistic, is the thing that is supposed to charge for having chosen it. The design that might have made the theorem true without destroying the mint is to keep the evidence and charge the selection on the price side, adding the seed pair's own naming cost. That design has now been run and it does not work, and the way it fails identifies the defect exactly.

The charge is affordable. At its natural size, 5.3 to 12.2 bits, both synthetic controls keep their exact node count and a perfect index under the canonical order and over twenty arrival orders, the deferred arm of E7 keeps its two nodes with identical supports and no typo node on all five seeds, no node is minted on any of the 48 null streams nor on any of the 300 wider ones, and the Wikipedia demonstration improves, its twenty-order mean index rising from 0.8438 to 0.8713. A uniform slack of 256 bits still leaves both synthetic controls exact. And it is ineffective: the exceedance stays 1.000 at every level, alone and combined with the tight naming charge.

Those two facts together are the diagnosis. A naming charge is constant in the stream length by construction, and the deficit is not: under the null the accumulator drifts at 1.42 bits per accrual step at $T = 500$ and 2.76 at $T = 4{,}000$, so the slack that would restore the nominal level at $b = 8$ is 97 bits at the first length and $1{,}205$ at the last. The charge of Theorem 1(iii) is a correct multiple-comparisons correction, and it is being asked to pay for something that is not a multiplicity. The corroboration is that with the seed key's own term removed from the statistic the residual requirement is 3.4 to 3.6 bits and flat in T, which is what a multiplicity looks like, and the same 5.32 bit charge covers it, holding at all five levels; that combination is the only arm in this series that satisfies the bound, and it is not shippable because removing the seed term destroys the mint.

One qualification the charge itself needs. Written as the log of the key count plus the log of that key's alphabet, it is Kraft-tight for a closed schema only. Under the open alphabet the escape mechanism creates, naming each candidate with the counts current at its own seed time is not a prefix code over a candidate set that grows with the stream. A universal integer code on the seed pair's interning ranks is Kraft over the unbounded set and still constant in T, and is the form to use.

Justification of the implemented statistic. The implemented escrow differs from the full likelihood ratio in one way, and this argument shows the difference is paid for. Validity requires the escrow to be computed on the full stream, never on a post-selection subsequence: conditioning on per-record attachment counts breaks the complete randomness and the exchangeability the martingale rests on (Doshi-Velez & Williamson, 2015). The implemented escrow accrues only on records carrying the pattern and drops the absence terms $\log(1 - m_p/u)$ elsewhere. Dropping these negative terms can only inflate $G_t(p)$, by at most $O(m_p \log(t/s))$ bits over the escrow window, so the omission by itself points in the anti-conservative direction. The guarantee of Theorem 1(ii) holds for the full ratio, and the omission is paid for once that bound is charged against the candidate. What this argument does not repair, and what the exceedance above measures, is the accrual index set: the price charges for the dropped absence terms, not for the choice of the records on which evidence is accrued at all. The implemented release price does exactly that: it carries the cohort-membership bits, the $\log \binom{n}{t}$ leading term of the column code. It is the same allocation accounting, relocated to the price side. □

C.4 THE IDENTITIES BEHIND THE TYPED EDGES

Every decision is emitted with a receipt: typed edges naming which attributes decided it, and at what price in bits. Three results say why those receipts can be trusted. Separability says the per-attribute decomposition is exact, not a narrative readout. The total-correlation identity says what quantity the escrow is actually metering. Consistency says the whole procedure needs no significance level.

A candidate must declare a scope $S \subseteq A$: the set of attributes it claims to say something about. Is choosing the best scope a combinatorial search? Under the scope code we use, no: the best scope falls out of independent per-attribute tests. In the lemma, g_a is the accumulated per-attribute saving, the bits attribute a has earned for the candidate so far; $\ell_{\mathrm{in}}(a)$ and $\ell_{\mathrm{out}}(a)$ are the costs of declaring a inside or outside the scope, with Bernoulli-KT estimators fitted to past scopes; $\Delta\ell_a = \ell_{\mathrm{in}}(a) - \ell_{\mathrm{out}}(a)$ is the marginal cost of including a; and $[x]_+$ means $\max(x, 0)$, the positive part.

Lemma C.2 (separability, with converse)**.** *Under a factorised scope code,* $L_{\text{scope}}(S) = \sum_{a\in S}\ell_{\text{in}}(a) + \sum_{a\notin S}\ell_{\text{out}}(a)$*, and*

$$\max_{S\subseteq A}\Big[\sum_{a\in S} g_a - L_{\text{scope}}(S)\Big] \;=\; \sum_{a\in A}\big[g_a - \Delta\ell_a\big]_+ \;-\; \sum_{a\in A}\ell_{\text{out}}(a),$$

attained by $S^* = \{a : g_a > \Delta\ell_a\}$*: independent per-attribute tests, an accept or reject per attribute, not an argmax over subsets. Conversely, under the combinatorial code* $\log\binom{|A|}{|S|}$ *plus* $\log(|A|{+}1)$*, the marginal price of the* s*-th included attribute is* $\log\frac{|A|-s+1}{s}$*, strictly decreasing in* s*, and the optimum becomes sort-by-*g_a*-and-cut: tractable, no longer separable. Separability is a property of the model class (facet-factorised emissions, factorised scope code), not a theorem about the data.*

The converse earns its place by showing the property is fragile: under the combinatorial code the per-attribute receipts are no longer exact. The shipped support code is the combinatorial one, $L_{\mathbb{N}}(|T|) + \log\binom{A_n}{|T|}$ (Table 2), so the shipped receipts are the sorted-prefix optimum, and per-attribute exactness is a property of the factorised alternative rather than of the shipped code.

Lemma C.2 states that under a factorised scope code the best scope is found by independent per-attribute tests, and, conversely, that a combinatorial scope code couples those decisions. The proof is a rearrangement in each direction.

Proof. For the direct claim, rearrange the objective: $\sum_{a\in S} g_a - L_{\text{scope}}(S) = \sum_{a\in S}(g_a - \Delta\ell_a) - \sum_{a\in A}\ell_{\text{out}}(a)$. The second term is constant in S. The first is maximised by including exactly the attributes with $g_a > \Delta\ell_a$, each contributing its positive part. So the maximiser is the accept set $S^* = \{a : g_a > \Delta\ell_a\}$, and the maximisation separates into independent per-attribute tests.

For the converse, the combinatorial scope code prices the next inclusion by $\log\binom{|A|}{s} - \log\binom{|A|}{s-1} = \log\frac{|A|-s+1}{s}$, so the charge for the next included attribute depends on how many are already included, coupling the decisions. For each size k the best scope is the k largest gains, so the optimum is always a prefix of the sorted order, found by scanning the cut point. Both the sorted gains and the marginal price fall, so the first crossing need not be the argmax, and the scan runs over all k. Separability is therefore a property of the model class (facet-factorised node emissions and a factorised scope code), not a theorem about the data. □

What does the escrow actually measure? Total correlation: the number of bits per record lost by modelling attributes independently when a latent distinction couples them. It is the multivariate form of mutual information, and it is zero exactly when the attributes are independent given the node. The other quantity in the proposition is the Jensen-Shannon divergence $\text{JS}_\pi(p, q)$: the mutual information between one observation and which of the two distributions produced it, a measure of how distinguishable p and q are, in bits per observation.

Proposition C.2 (total correlation funds the mint)**.** *Let nodes* z_i, z_j *carry facet-product profiles* $P_i = \prod_a p_i(\cdot \mid a)$ *and* $P_j = \prod_a p_j(\cdot \mid a)$*, merged with weights* (π_i, π_j) *into* $\bar{z}$*. Let* Y *be a record drawn from the merged node,* Y_a *its value on attribute* a*, and* S *the component indicator recording which of the two profiles produced it. With* $\text{JS}_a = \text{JS}_\pi(p_i(\cdot \mid a), p_j(\cdot \mid a)) = I(Y_a; S)$ *the per-attribute divergence,* $\text{JS}_{\text{joint}} = I(Y; S)$ *the joint one, and* $\text{TC}(\bar{z}) = \sum_a H(Y_a \mid \bar{z}) - H(Y \mid \bar{z})$ *the total correlation of the merged node,*

$$\textstyle\sum_a \text{JS}_a \;=\; \text{JS}_{\text{joint}} + \text{TC}(\bar{z}), \qquad \text{TC}(\bar{z}) \geq 0,$$

with equality iff the profiles differ in at most one attribute. In the facet-product code a merge raises the data bits by exactly $\bar{p}\sum_a \text{JS}_a$ *per record. The joint criterion does not decompose per attribute in general; the sum overcounts* $I(Y; S)$ *by the inter-attribute redundancy about the erased distinction. But in this model class the per-attribute sum is not an approximation, it is the exact price.* TC *is the overprice the product code pays for refusing to represent a latent distinction, and the escrow accumulates it: over a cohort of* n_s *records the accumulated saving is* $n_s[\sum_a \text{JS}_a - H(\pi)]$ *up to universal-code regret* $O(\log t)$*, and since* $0 \leq H(\pi) - \text{JS}_{\text{joint}} \leq H(\pi) \leq 1$ *bit, the escrow is a total-correlation meter to within one bit per cohort record: a node is minted exactly when several facets redundantly witness one latent distinction long enough to pay* Λ_s*.*

The proof rests on the mixture identity for the Jensen-Shannon divergence (Lin, 1991).

Proposition C.2 states that the per-attribute Jensen-Shannon savings sum to the joint saving plus the total correlation of the merged node, so the escrow is a total-correlation meter. Two quantities carry the proof. The Jensen-Shannon divergence of a two-component mixture is the information the observation carries about which component produced it. The total correlation of a node is the information its attributes share with one another: zero exactly when they are independent given the node.

Proof. The starting identity is $\mathrm{JS}_\pi(P, Q) = I(Y; S)$ for the mixture with component indicator S (Lin, 1991). Given S the attributes are independent, because each component is a product, so $H(Y \mid S) = \sum_a H(Y_a \mid S)$. Then $\sum_a \mathrm{JS}_a = \sum_a [H(Y_a \mid \bar{z}) - H(Y_a \mid S)]$ and $\mathrm{JS}_{\mathrm{joint}} = H(Y \mid \bar{z}) - \sum_a H(Y_a \mid S)$. Subtracting the second from the first gives the identity (verified to machine precision), and $\mathrm{TC} \geq 0$ is subadditivity of entropy.

For the equality condition: if only facet b differs between the components, the mixture equals the product of the shared facets with the mixed facet b, hence $\mathrm{TC} = 0$. If facets $a \neq b$ both differ, then $P(y_a, y_b) - P(y_a)P(y_b) = \pi_i \pi_j \big(p_i(y_a) - p_j(y_a)\big)\big(p_i(y_b) - p_j(y_b)\big) \not\equiv 0$, so $\mathrm{TC}(\bar{z}) \geq I(Y_a; Y_b) > 0$ by the grouping bound.

The escrow reading follows from what each side pays. The incumbent that refuses the candidate codes the cohort with the merged product, at cost $\sum_a H(Y_a \mid \bar{z})$ per record. The candidate restores the distinction, at allocation cost $H(\pi)$ per record. Krichevsky-Trofimov block codes attain these entropies to $O(\log t)$ regret (Krichevsky & Trofimov, 1981). □

The exchange rate here is derived, not chosen. The information bottleneck (IB) family trades compression against preserved information at a user-chosen rate β: how many bits of code one bit of kept information is worth. In our code the rate is pinned. Transmitting the node name and then the record given the node costs $H(T) + H(Y) - I(T; Y)$ per record, the deterministic information-bottleneck objective at exactly $\beta = 1$, the slope minus-one point where one bit of distortion costs one bit of code. At that exchange rate many clusterings tie exactly, so the IB objective alone cannot choose among them, and the only remaining term that orders the candidates is the model cost. In IB language: at that slope the population frontier is an entire indifference face, so the price Λ_s is not a regulariser bolted onto an IB objective: it is the whole selection principle once β is pinned. Agglomerative IB computes but never spends its own rate decrease and runs to one cluster on an elbow heuristic (Slonim & Tishby, 1999); DIBmix retunes β every iteration and still needs a knee criterion and a user-fixed cluster count (Costa et al., 2026). The IB family externalises both the exchange rate and the stop; the prequential code internalises both.

Deferral would be worthless if the escrow released for noise, or never released at all. The next proposition closes both directions, with no significance level anywhere. The classical input is Wilks' theorem: on data that truly follow the incumbent law, the maximised log-likelihood-ratio gain of an alternative with d_S free parameters is asymptotically a chi-squared random variable with d_S degrees of freedom, so the gain from chance stays bounded in probability while every charge in the escrow keeps growing. Bounded in probability is written $O_P(1)$ below. The growing charge has a name too: $\mathrm{COMP}(d_S, n)$ is the NML parametric complexity of Appendix D, here for a scope carrying d_S free parameters, the regret the candidate's own growing blocks pay, inside G_n, relative to the best fixed distribution chosen in hindsight.

Proposition C.3 (consistency without a significance level)**.** *Let a candidate's scope carry* $d_S = \sum_{a \in S}(d_a - 1)$ *free parameters. (i)* No spurious nodes. *If the cohort's values follow the incumbent law on every scope attribute, the empirical gain after* n *cohort records is* $\chi^2_{d_S}/(2 \ln 2) = O_P(1)$ *by Wilks, with mean* $d_S/(2 \ln 2) \approx 0.721\, d_S$ *bits, a constant, while the charge inside the escrow grows as* $\mathrm{COMP}(d_S, n) = \frac{d_S}{2} \log \frac{n}{2\pi} + O(1) \to \infty$*, and the frozen* $\Lambda_s > 0$ *must be cleared on top. Hence* $G_n \to -\infty$ *in probability, and the probability that a spurious candidate stands at or above its charge at cohort size* n *tends to zero, with no significance level anywhere; the lifetime guarantee over the running supremum is Theorem 1's. Comparing means: pure noise stops being able to clear the parameter charge once the cohort exceeds* $2\pi e \approx 17$ *records. (ii)* No missed structure. *If a latent feature activates a fraction* $\rho > 0$ *of the stream and differs from the incumbent by* $D > 0$ *bits per record on its scope, the evidence grows as* $\Theta(\rho D n)$ *against a* $\Theta(\log n)$ *charge, so the node is*

minted almost surely, at cohort support $n^* = \Theta\big(\frac{d_S}{2D} \log \frac{d_S}{2D}\big)$*. If* $D = 0$*, the expected crossing time is infinite and the lifetime mint probability is at most* $2^{-\Lambda_s}$*: in expectation the candidate is never minted, which is correct behaviour. Together with Lemma 1, both degeneracies close: one node per record is excluded by the validity of the code, one node total by (ii).*

Proposition C.3 states that a spurious candidate is eventually refused and a real one is eventually minted, with no significance level anywhere. The proof compares two growth rates: the evidence a cohort accumulates, against the charge its free parameters accrue.

Proof. (i) The candidate's block code is the KT mixture, whose regret against the cohort's empirical distribution is, for the d_a-ary facet, $\frac{d_a-1}{2} \log \frac{n}{2\pi} + \log \frac{\pi^{d_a/2}}{\Gamma(d_a/2)} + o(1)$ (Xie & Barron, 2000). Summed over the scope this is $\mathrm{COMP}(d_S, n)$, the parametric complexity: the total charge for the candidate's d_S free parameters. So G on the cohort equals $n\,\mathrm{KL}_2(\hat{p} \,\|\, q) - \mathrm{COMP}(d_S, n) + O(1)$. Under the null, Wilks' theorem says the multinomial likelihood ratio against the fixed incumbent converges to a $\chi^2_{d_S}$ limit; the estimation error of the incumbent's own predictive is lower order, since its cumulative redundancy is $O(\log n)$ on a cohort its tables dominate. So the gain is $O_P(1)$, with mean $d_S/(2 \ln 2) \approx 0.721\, d_S$ bits, while the charge diverges. That divergence is the sign of $d\,\mathrm{COMP}/dn$ doing the work a significance level does elsewhere. Equating means gives the crossover at cohort size $2\pi e \approx 17$ records, past which pure noise can no longer clear the parameter charge in expectation.

(ii) Supporting records arrive at rate ρ, so by time n the cohort has $\approx \rho n$ records and evidence $\approx \rho n D$ bits. That exceeds $\frac{d_S}{2} \log n + \Lambda_s = \Theta(\log n)$ for all large n, and solving $nD = \frac{d_S}{2} \log n + \Gamma$ for the crossing gives the stated order of n^*. Exact KT crossings at $\Gamma = 20$ bits: $d_a = 2$, $D = 1$ mints at 24 cohort records; $d_a = 100$, $D = 1$ at 199; $D = 10$ at 3 to 5 across arities. If $D = 0$ the expected crossing time is infinite and the lifetime mint probability is at most $2^{-\Lambda_s}$ by Theorem 1. One caution on the expansion: the asymptotic form of COMP is for the analysis only. The algorithm, and the crossings quoted, use the exact Gamma-function form, since $\mathrm{COMP}(m, n)$ goes negative for $n \ll m$. □

The crossover quoted in Section 4 is this closed form, $2\pi e \approx 17.08$ records, and it is arity-free only at leading order: keeping the term the expansion drops puts the asymptotic crossover at the alphabet size itself, and the exact Gamma-function form puts it at 1, 2, 7 and 35 records at alphabet sizes 2, 5, 20 and 100 (`results/e15_theory_numbers.json`). The proposition is the fixed-alphabet idealisation of the shipped open-alphabet code, whose blocks use the escape of Appendix D rather than an explicit complexity charge.

C.5 The repair pass

The streaming rule is irrevocable, and the repair pass of Section 3 revisits its decisions. Why can the repair not simply keep minimising the streaming code? Because the prequential code's past is spent: each record was priced by the predictive that stood at its arrival, and those prices cannot be re-issued, so "monotone repair on the prequential code" is meaningless. A repair pass needs a second objective that is a function of the current state alone, and order invariance forces its shape: an exchangeable allocation probability sees only column statistics, so an order-invariant price is a function of node size, while the streaming price is a function of birth position, and the two cannot appear in one functional.

A state is $G = (K, Z, S, \omega)$: K nodes; an allocation matrix $Z \in \{0,1\}^{n \times K}$ with column sums m_k, multi-membership allowed; supports $S_v \subseteq A$ over the declared key universe of size $|A|$; and an ownership map ω assigning every published (record, key) cell to exactly one supporting member node or the background, so no cell is coded twice. Write $p_{v,a}$ for the members of v publishing a, and $h_{v,a}$, $h_{0,a}$ for the owned and background value counts. The batch objective $L_{\mathrm{batch}}(G)$ of Equation (C.7) is the total codelength of this state, five sums in order: a universal integer code on $K+1$ (Rissanen, 1983), so the empty state stays codable; a scope code per node; the negative log IBP feature-allocation probability at mass γ (Griffiths & Ghahramani, 2011); per-facet value and presence blocks, KT for the owned counts and exact normalized maximum likelihood for the block counts, computed by the Kontkanen-Myllymäki recursion (Kontkanen & Myllymäki, 2007), within

$O(1)$ of KT (Xie & Barron, 2000); and the background blocks. The node-dependent presence blocks are load-bearing: without them nothing opposes merging nodes with disjoint supports (recomputed from the shipped objective at $|A| = 200$, $n = 10^4$: such a merge of two 50-member nodes scores $+289.27$ bits with the term and -110.73 without, and every pair of an eight-node disjoint-support graph pays to merge without it, although the shipped repair proposes no such merge, so the collapse is a property of the objective rather than a state the implementation reaches).

In full, L_{batch} is one displayed equation, the five sums above with every symbol named: Equation (C.7) in Appendix C.6.

Proposition C.4 (order-invariant repair, and its termination)**.** *(i) L_{batch} contains no arrival index: it is a function of the counts $\big(K, \{|S_v|\}, \{m_k\}, \{p_{v,a}\}, \{h_{v,a}\}, \{h_{0,a}\}, n\big)$ alone and is invariant under every permutation of the records; no birth position and no timestamp occurs anywhere in it. (ii) Let the repair pass propose* MERGE, SPLIT, REASSIGN, *and* DELETE *moves, each preserving the invariant that every node owns at least one published cell, and accept a move exactly when it strictly decreases L_{batch}, ties rejected. Then the pass performs finitely many accepted moves and terminates at a local minimum of L_{batch} with respect to the repertoire.*

Proof. Part (i) is Lemma C.3 and part (ii) is Proposition C.5, both stated and proved in Appendix C.6 below on the closed form of Equation (C.7). □

Part (ii) is the analogue, for the repair pass alone, of Theorem 3.1 of DP-means (Kulis & Jordan, 2012), which proves monotone decrease for a batch multi-pass algorithm and nothing about a per-record streaming rule. As there, termination does not imply agreement across arrival orders, which E5 measures (Section 5.2: exact on planted structure, ARI 0.6389 to 0.986 on Wikipedia), and the composite of stream plus repair is monotone on no single functional.

C.6 THE BATCH OBJECTIVE IN FULL

The streaming rule minimises the prequential code: the bits actually spent, arrival by arrival, on a past that is immutable once transmitted. That quantity is monotonically increasing by construction, so "monotone repair on the prequential code" is not unproven, it is meaningless. A repair pass needs a second objective, stated once, that is a function of the current state alone. The two objectives cannot be one. The streaming price of a node is a function of its birth position, which is what makes it computable online. An order-invariant price must instead be a function of node size, because an exchangeable allocation probability sees the allocation only through its column statistics. The two cannot appear in one functional. We therefore state the two-objective structure plainly. The streaming rule minimises the prequential code, whose past is spent. The repair pass minimises the batch objective L_{batch}, which is order-invariant. The two agree to the standard $O(\log n)$ per parameter prequential-versus-two-part equivalence (Grünwald, 2007; Dawid, 1984). And the realised gap between them is a measured quantity (Definition C.2), never an assumption.

The batch objective in closed form. In full,

$$\begin{aligned} L_{\mathrm{batch}}(G) = L_{\mathbb{N}}(K{+}1) + \sum_{v} \Big[\log|A| + \log \tbinom{|A|}{|S_v|} \Big] + L_{\mathrm{alloc}}(Z) \\ + \sum_{v} \sum_{a \in S_v} \big[L_{\mathrm{KT}}(p_{v,a}, m_v) + L_{\mathrm{NML}}(h_{v,a}) \big] + \sum_{a} L_{\mathrm{NML}}(h_{0,a}), \end{aligned} \tag{C.7}$$

with $L_{\mathbb{N}}$ the universal integer code (Rissanen, 1983), L_{alloc} the negative log IBP feature-allocation probability at mass γ (Griffiths & Ghahramani, 2011) (per-column factor $\binom{n}{m_k}^{-1} m_k^{-1}$, a column-multiplicity term, mass terms $\gamma H_n \log e - K \log \gamma$), L_{KT} the KT block code, and L_{NML} the normalized maximum-likelihood codelength, exact by the Kontkanen-Myllymäki recursion (Kontkanen & Myllymäki, 2007), within $O(1)$ of KT (Xie & Barron, 2000). The definition below gives the shipped instantiation, which replaces the exchangeable allocation factor L_{alloc} by the declared γ-free convention.

Definition C.1 (the batch repair objective, shipped instantiation)**.** *A state is $G = (K, Z, S, \omega)$: K latent nodes; an allocation $Z \in \{0,1\}^{n \times K}$ with column sums m_k (multi-membership allowed); supports $S_v \subseteq A$; and an ownership map ω assigning every published (record, key) cell to exactly*

one member node supporting that key, or to the background, so that no cell is coded twice. L_{batch} is the sum of the terms E1, E2, E3, E4, E6, E7, E8, E9 and E11 of Appendix D, evaluated on the count statistics of G: the universal-integer header and label-invariance term (E1), one Krichevsky-Trofimov column code per node (E2), the support codes (E3), the node-dependent presence blocks (E4), the owner-index bits (E6), and the value blocks of the nodes and the background (E7 to E9). This is the declared γ-free convention of Lemma C.4: the exchangeable IBP allocation factor is replaced by the header plus the column codes, a function of (m_k, n) alone, within $\frac{1}{2}\log n + O(1)$ bits per node of the exchangeable feature probability function factor (Griffiths & Ghahramani, 2011; Broderick et al., 2013b).

Lemma C.3 (order erasure)**.** *L_{batch} contains no arrival index: it is a function of $\big(K, \{|S_v|\}, \{m_k\}, \{p_{v,a}\}, \{h_{v,a}\}, \{h_{0,a}\}, n\big)$ alone. Consequently, for every permutation σ of $[n]$, relabelling the records by σ leaves L_{batch} unchanged. In particular no birth position and no per-record timestamp occurs anywhere in the objective.*

Proof. By inspection of Definition C.1 and the term list of Appendix D: every term depends on Z only through its column sums, and on ω only through the count vectors p and h. Row permutations of Z and relabellings of cells preserve all of these, and n enters only as the size of the record multiset. A batch objective that keeps a birth position inside it fails exactly here: the same state reached by two histories carries two different values, strict decrease no longer forbids revisiting a state, and termination is unprovable. The relabelling half of this invariance is asserted by `code/tests/test_batch.py` (difference 0.0 over 20 relabellings of the nodes); gate UT-15 in its 200-record-permutation form remains designed, and two arrival orders that reach the same member partition can still differ by up to 225 bits through cell ownership, so the invariance is of the objective in the state, not of the state in the order. □

The node-dependent presence blocks (E4) are load-bearing in this objective. For the insertion decision a node-independent missingness code cancels from every comparison, a genuine simplification there. Inside L_{batch} the same cancellation removes the only term opposing the merge of two nodes with disjoint supports. Recomputed from the shipped objective at $|A| = 200$, $n = 10^4$, a disjoint-support merge of two 50-member, two-key nodes scores $+289.27$ bits with the node-dependent presence term and -110.73 without it; at $t = 200$ with four keys, $+2778.26$ against -421.74; at $t = 1000$ with four keys, $+13822.96$ against -2177.04. The three differences are exactly $4tk$ bits, which is what the term contributes, and they sit within 7 bits of the values recorded during development. On an eight-node disjoint-support graph all 28 pairs pay to merge once the term is removed and none pays with it, but the shipped repair proposes a merge only for nodes that share a support key, so a full pass applies no move and K stays 8 under either arm: the collapse is a property of the objective's sign, not a state the implementation reaches. The lesson is that non-degeneracy proved for the insertion rule does not transfer to the repair operators. With the guard it is re-established for merge, and the delete move is the corresponding guard in the opposite direction.

Proposition C.5 (monotone termination of the repair pass)**.** *Fix the record multiset. Let the repair pass propose moves from the repertoire $\{\mathrm{MERGE}, \mathrm{SPLIT}, \mathrm{REASSIGN}, \mathrm{DELETE}\}$, each mapping a state to a state for the same records and preserving the invariant that every node owns at least one published cell (a node stripped of its last cell is deleted within the same move, so $m_k \geq 1$ and every allocation factor is defined). Let a move be accepted exactly when it strictly decreases L_{batch}; ties are rejected. Then the pass performs finitely many accepted moves and terminates at a state where no move in the repertoire decreases L_{batch}: a local minimum of the objective with respect to the repertoire.*

Proof. Three facts. (i) Bounded below: every term of L_{batch} is the negative logarithm of a (sub)probability, so $L_{\mathrm{batch}} \geq 0$. (ii) Finite state space: by the invariant each node owns one of at most $n|A|$ published cells, so $K \leq n|A|$; a state is then an allocation in $\{0,1\}^{n\times K}$, supports drawn from the $2^{|A|}$ subsets of A, and an ownership map over at most $n|A|$ cells with at most $K+1$ labels, and for fixed n and $|A|$ there are finitely many such states. (iii) Strict descent: the accepted states carry strictly decreasing objective values, so no state recurs, and a strictly decreasing sequence on a finite set is finite. At termination every candidate move has $\Delta L_{\mathrm{batch}} \geq 0$, which is the definition of a local minimum with respect to the repertoire. □

Scope, held together in three sentences. This proposition is about the repair pass only. The stream is not monotone, because the prequential past is spent. The composite of stream plus repair is monotone on no single functional. The proposition is the repair-pass analogue of Theorem 3.1 of DP-means (Kulis & Jordan, 2012), which proves monotone decrease per complete iteration of a batch multi-pass algorithm and proves nothing about a per-record streaming rule. And since Kulis and Jordan's own paper concedes dependence on the processing order, termination does not imply agreement across arrival orders, there or here. That agreement is measured by E5 (Section 5.2), never assumed. The guarantee is finiteness, not a rate: the implementation caps the number of full passes and reports whether the cap binds.

One more fact is needed before the gap can be defined. The batch objective inherits the IBP mass γ, and the next lemma settles what can and cannot be done with it.

Lemma C.4 (the batch mass is irreducible)**.** *In the exchangeable-allocation form of the batch objective the IBP mass γ cannot be optimised away. (i) Profiling fails: substituting $\hat{\gamma} = K/H_n$ turns the two mass terms into $K[1 + \ln(H_n/K)] \log e$ bits, decreasing in K for $K > H_n$ and unbounded below, so the profiled objective rewards nodes instead of pricing them; and the profiled assignment is not a code, because maximised likelihoods do not satisfy the Kraft inequality: the Shtarkov normaliser of the family is $\sum_K K^K e^{-K}/K!$, which diverges (Shtarkov, 1987), so no normalized maximum-likelihood code for the allocation exists at any n. (ii) Improper mixing fails: the Fisher information is $I(\gamma) = H_n/\gamma$, so Jeffreys' prior is $\propto \gamma^{-1/2}$ and improper; every improper mixture has infinite total mass, hence is not a code, and at scale $b = 0$ the first-record predictive assigns probability zero to every finite number of features. (iii) What remains is relocation with a receipt: a proper $\mathrm{Gamma}(a, b)$ mixture is a valid code and demotes γ to the hyperprior (a, b), with shape $a = \frac{1}{2}$ Jeffreys-canonical and scale $b > 0$ forced positive by Kraft, with total influence $O(1)$ bits over the stream and per-decision influence $O(1/\log n)$ (along $K_s = 2H_s$: 0.69 bits per mint at $n = 10$, 0.19 at $n = 10^5$); or γ is eliminated syntactically by the universal allocation code of Definition C.1, at a redundancy of $\frac{1}{2}\log n + O(1)$ per node. In every case the freedom is demoted and disclosed, never derived away.*

Proof. (i) The stationary point of $-K \ln \gamma + \gamma H_n$ is $\hat{\gamma} = K/H_n$, with value $K[1 + \ln(H_n/K)]$ nats and derivative $\ln(H_n/K)$ in K. For the normaliser, $\sup_\gamma \gamma^K e^{-\gamma H_n} = (K/H_n)^K e^{-K}$, and the γ-free factors summed over allocations with K columns give $H_n^K/K!$, since the exchangeable form must reproduce the $\mathrm{Poisson}(\gamma H_n)$ marginal; the product is $K^K e^{-K}/K!$, and Stirling gives the divergence rate (partial sums grow as $\sqrt{2K/\pi}$, verified numerically). (ii) is immediate from total mass, and the $b = 0$ statement from $\mathbb{E}[K] = a/b$. (iii) The mixture codelength differences between proper hyperpriors converge to the log ratio of the priors at the posterior mean; the measured values are quoted in the statement. □

Lemma C.4 is why the repair pass does not inherit the claim of no calibrated parameter wholesale: it descends an objective whose mass freedom is demoted to a declared convention, published with its sensitivity. The prequential spine escapes the same vise for a structural reason. In the batch functional the mass multiplies the node count, so a factor-two change of γ moves every create-or-destroy margin by exactly one bit per node and can move the minimiser. In the online code the mass enters each decision once, through the mint price frozen at seed time. Under the deferred mint that price is a stopping boundary rather than a per-record comparison. Residual freedom in the convention therefore perturbs when a candidate is minted, not whether: any candidate with a positive evidence rate crosses every boundary in the family $c \log n$, $c \in [1, 2]$. The same Kraft argument that kills batch profiling forces the online hyperprior to be proper; the two halves of the method fail and escape for the same reason, and that is the strongest argument we know for the prequential form.

Definition C.2 (the one-pass-to-batch gap)**.** *Fix an arrival order σ. Let $G_n^{\mathrm{stream}}(\sigma)$ be the state built by the one-pass rule and $G_n^{\mathrm{rep}}(\sigma)$ the terminal state of the repair pass started from it. The repair gap is $\Delta_{\mathrm{rep}}(n, \sigma) = L_{\mathrm{batch}}(G_n^{\mathrm{stream}}(\sigma)) - L_{\mathrm{batch}}(G_n^{\mathrm{rep}}(\sigma)) \geq 0$, nonnegative by Proposition C.5, to be reported in bits and in bits per record; on synthetic streams the oracle gap against the planted allocation is to be reported beside it.*

Δ_{rep} is computable on every run with no ground truth, which is what makes it reportable, and on the twenty-node stream below it is 9,966 bits, 0.50 bits per record (`results/e15_theory_numbers.json`); the shipped runners do not yet record it per run.

It is what "approximation gap of the one-pass rule" means everywhere in this paper. For scale, on a 20,000-record stream with 20 planted nodes the myopic rule mints nothing, so its state is exactly the all-background code, and that code is $198{,}478$ bits longer than the planted allocation, 9.92 bits per record and 31.57 percent of its own code; the deferred rule recovers all twenty nodes and lands $4{,}176$ bits below the planted allocation, because it owns cells the planted labelling does not.

D THE ENCODING, TERM BY TERM

Terms used here: KT is the Krichevsky-Trofimov code and NML normalized maximum likelihood, both in Appendix A; a block is the set of counts a node keeps for one key; an escape is the part of a code that prices a value never seen before under a key; E1 to E12 number the terms of the released specification, and UT-n names a gate of Appendix E.

Two codelength objects appear in this paper, and they are not the same object. L_{emit} is the prequential codelength actually emitted, arrival by arrival: order-dependent, and spent. L_{batch} is the order-invariant two-part codelength of the same prefix, and it is the objective. Insertion is a greedy step on it. Every repair move is an exact evaluation of it. The gap between the two is measured, never assumed (Definition C.2). The batch objective is the sum E1+E2+E3+E4+E6+E7+E8+E9+E11; E10 is a statement about what the fresh-node reference is, and E12 is the composite mint price assembled from the others. The numbering follows the specification. Every predictive sums to exactly one over its alphabet including the escape symbol; nothing is normalised by a quantity the decoder cannot compute from the past. In the table, KT abbreviates Krichevsky-Trofimov, the smoothed-count code introduced in Appendix C.

Term	Object	Closed form	Source
E1	node count	$L_{\mathbb{N}}(K) = \log^* K + \log c_0$, $c_0 = 2.865064$; plus $-\log K!$ for label invariance	(Rissanen, 1983; Galbrun, 2022)
E2	membership column	$L_{\mathrm{col}}(t, n)$, the KT Beta($\frac{1}{2}, \frac{1}{2}$) mixture over binary columns	(Krichevsky & Trofimov, 1981; Willems et al., 1995)
E3	support	$L_{\mathbb{N}}(\lvert T\rvert) + \log \binom{A_n}{\lvert T\rvert}$, A_n = keys seen so far	(Galbrun, 2022; Vreeken et al., 2011)
E4	presence	per $(v, a \in S_v)$: KT Bernoulli on $(p_{v,a},\ t_v - p_{v,a})$; background Bernoulli elsewhere	(Krichevsky & Trofimov, 1981)
E6	owner index	$\log(1 + q(a))$ over the competing owners plus background	(Vreeken et al., 2011; Galbrun, 2022)
E7	categorical values	KT sequence code $L_{\mathrm{KT}}(\mathrm{counts}, d)$; predictive $-\log \frac{c+1/2}{N+d/2}$; three-stage escape for open alphabets	(Krichevsky & Trofimov, 1981; Xie & Barron, 2000; Cleary & Witten, 1984)
E8	numeric values	Jeffreys normal-inverse-gamma Student-t predictive plus $\log(1/\delta_a)$; two-expert $\{x, \log x\}$ mixture	(Grünwald, 2007)
E9	free text	prequential token-level KT with PPM-style escape; two-expert tokenisation mixture; quarantined in v1	(Cleary & Witten, 1984)
E10	the reference	no separate fresh-node reference code exists: the alternative owner is another node or the background, itself a real E7 to E9 block	
E11	announcements	KT Bernoulli over the edit history; 2 bits for the move type; $2 \log K$ for operands	(Willems et al., 1995)
E12	the mint price	$\mathrm{Price}(t, T, n) = L_{\mathrm{ann}}(n) + 2 + L_{\mathrm{col}}(t, n) + L_{\mathrm{supp}}(T) + \Delta L_{\mathbb{N}} + \log(K{+}1)$	

Table 2: The encoding, term by term. There is no E5 in the specification; the numbering is preserved so that every term here matches its entry in the released specification and test suite. E12 satisfies $\sum 2^{-\mathrm{Price}} \le 1$ over (n, S, T) because each component is Kraft-tight (gate UT-4).

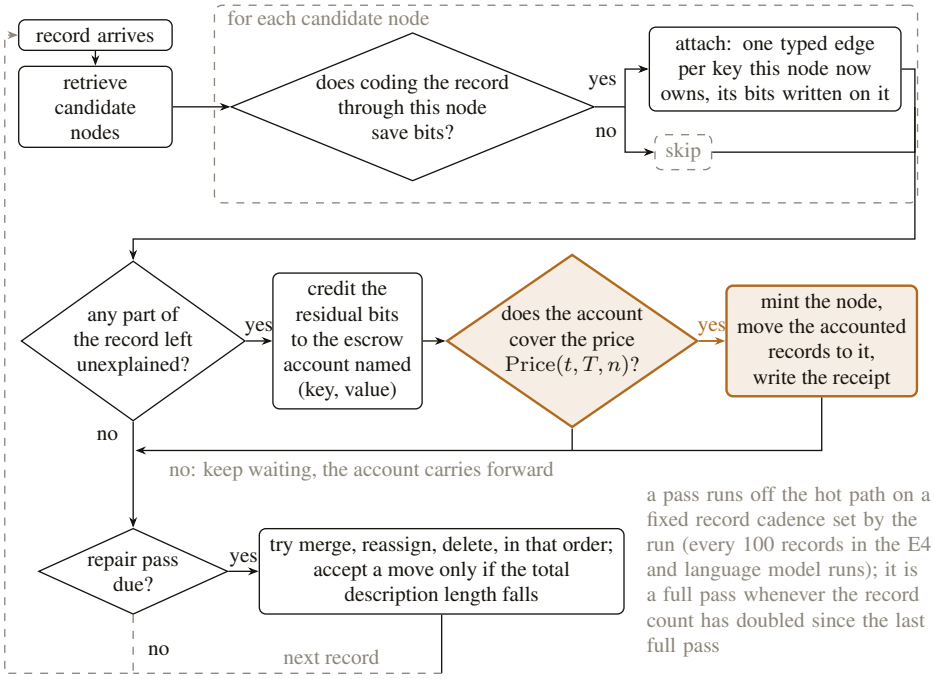


Figure 6: The decision for one record, and the repair loop. Diamonds are decisions taken in bits; the gold diamond and box are the release test of Equation 2 and the mint. The repair pass runs on a fixed record cadence that belongs to the run protocol, every 100 records under the protocol of Section 5.1, and is a full pass whenever the record count has doubled since the last full pass (code/escrow/engine.py and code/escrow/batch.py).

D.1 The online step in formulas

This subsection gives the three displayed formulas of the per-record step of Section 3, with the measurements that shaped each. KT abbreviates the Krichevsky-Trofimov code, the smoothed-count code for a sequence of symbols whose frequencies are unknown.

Figure 6 walks the loop. A record arrives and candidate nodes are retrieved by index, never by a scan of all nodes. For each candidate the engine asks whether coding the record through this node saves bits; if it does, the node is attached and one typed edge is emitted per key the node now owns, carrying its bits, and if it does not, the node is skipped. Any part of the record that remains unexplained credits its residual bits to the escrow account named by its (key, value) pair. If that account now covers the price $\mathrm{Price}(t, T, n)$ of stating the node, the node is minted, the accounted records move to it and a receipt is written; otherwise the account carries forward and the engine keeps waiting. When a repair pass is due, the engine tries merge, reassign and delete in that order and accepts a move only if the total description length falls; the pass is a full pass whenever the record count has doubled since the last full pass. The engine then moves to the next record.

Per-facet evidence. How strongly does one published key argue for one candidate? Per candidate v and published key $a \in S_v$, the evidence is the smoothed pointwise mutual information

$$d_n(a, v) \;=\; \log_2 \frac{\hat{P}_{v,a}(x_{n,a})}{\hat{P}_{0,a}(x_{n,a})} \;+\; \log_2 \frac{\hat{P}^{\mathrm{pres}}_{v,a}(1)}{\hat{P}^{\mathrm{pres}}_{0,a}(1)}, \tag{D.8}$$

two log-ratios in bits: how much better the node predicts the record's value than the background does, plus how much better it predicts that the key is present at all. Positive means a argues for attachment. Keys in S_v that the record does not publish contribute the corresponding absence term, the silence bill read as evidence. Every predictive is snapshotted before the record updates anything.

The active set is built greedily. The coverage gain of adding v (its per-key improvements over the current owners, plus its absence terms) is monotone submodular: each further node helps less,

because the owners it must beat keep improving. The price ($\mathrm{attach}(t_v, n)$ plus the owner-index increments) is modular: it simply adds. Greedy on a submodular gain minus a modular price carries the distorted-greedy guarantee of Harshaw et al. (2019): the returned active set achieves gain minus price at least $(1 - 1/e)$ times the best possible set's gain, minus that set's full price. Ties break by birth order, never by hash order, and the loop stops at the first non-positive gain. The decomposition is exact, not narrative: the per-facet terms, absence terms and attach price sum to the scalar create-or-absorb verdict, and gate UT-5 (Appendix E) asserts the identity to 10^{-9} bits against two independent recomputations of L_{batch}.

Single ownership. A record may activate several nodes, and this multi-membership is exactly what breaks the additivity every partition-based code enjoys: if two nodes both claim the same facet, its bits are counted twice, and the sum is no longer a code. The repair is an arbitration map. The owner of each facet is the candidate with the strongest claim,

$$k_n(a) \;=\; \arg\max_{v \in S_{r_n}} d_n(a, v), \tag{D.9}$$

where the maximum runs over S_{r_n}, the record's active set, and $d_n(a, v)$ is Equation D.8: one owner per facet, in the tradition of the Krimp cover (Vreeken et al., 2011). The map does double duty. It validates the code, because each facet is transmitted under exactly one model and every sum stays additive with no double counting. And it *is* the output, because the winning facet is emitted as the typed edge $(a, x_{n,a}) \to k_n(a)$, carrying its own price $d_n(a, k_n(a))$ in bits. Ownership is not free: naming the owner among the $q(a)$ nodes competing for the facet costs $\log_2(1 + q(a))$ bits per facet, a realised bill we report. One circularity should be stated rather than smoothed over: additivity is bought by single ownership, and single ownership was chosen partly because it yields the additivity being claimed.

The mint price. Structural edits are part of the stream being transmitted, so an edit must announce itself: a KT coin over the edit history prices "an edit happens now" (write $L_{\mathrm{ann}}(n)$ for that price), then two bits name the edit type and $O(\log_2 K)$ bits name the operands. Adding up the terms, the full price of a node with support T and t members at time n is

$$\mathrm{Price}(t, T, n) \;=\; L_{\mathrm{ann}}(n) + 2 + L_{\mathrm{col}}(t, n) + L_{\mathrm{supp}}(T) + \big[L_N(K{+}1) - L_N(K)\big] + \log_2(K{+}1), \tag{D.10}$$

the announcement, the edit type, the membership column, the support, the increment of the node-count code, and the new node's name. This is a Kraft code over (time, membership, support) jointly, and the joint scope is the point. The leading term of L_{col} is $\log_2 \binom{n}{t}$, and here is what it buys. Which t of the n records form the cohort is one choice among $\binom{n}{t}$ possibilities, and searching over all of them multiplies the false-alarm probability by at most $\binom{n}{t}$; that multiplier is the union bound, the rule that the chance that any of many options fires is at most the sum of their chances. Charging $\log_2 \binom{n}{t}$ bits divides the false-mint budget by exactly the same factor, so the search and the charge cancel. The act of picking a candidate by looking at the data is already paid for inside the price. The node-count term $L_N(K) - \log_2 K!$ uses Rissanen's universal integer code (Rissanen, 1983), with a rebate because node labels are interchangeable; the marginal cost of the K-th node is about $1/(K \ln 2)$ bits (0.0014 bits at $K = 1000$), numerically irrelevant, which is worth saying because the choice that looks most arbitrary does not matter.

Measured failures that shaped the code. Three measurements are recorded here because each one changed the design. Dropping the naming stage of the value escape makes a thin block's escapes nearly free without ever identifying the value: a Kraft violation, and it manufactured evidence, minting 204 sibling nodes on a planted two-node control. Measuring escrow evidence against the background for facets a minted node already explains double-counts evidence and mints siblings of existing nodes: 332 nodes before the owner-incumbent rule on the same control, 2 after. And in the repair pass, a true node that the one-pass rule attached only partially loses to the background if deletion is evaluated before reassignment (deletion pays exactly -50.8 bits on the planted control), while the same node fully attached wins; this is why merges run first, then reassignment, and deletes last.

D.2 THE TERMS, ONE BY ONE

The announcement code follows the switching and expert-tracking line (Willems, 1996; Herbster & Warmuth, 1998); the numeric predictive is the standard Student-t (Bernardo & Smith, 1994); the two-expert mixtures use the aggregating framework of Cesa-Bianchi & Lugosi (2006).

E2, the term that replaces λ. The column code prices a node's membership column: which t of the n records so far belong to it. Its closed form is four log-Gamma terms,

$$L_{\mathrm{col}}(t,n) \;=\; \log_2 \pi + \log_2 \Gamma(n{+}1) - \log_2 \Gamma(t{+}\tfrac{1}{2}) - \log_2 \Gamma(n{-}t{+}\tfrac{1}{2}). \tag{D.11}$$

Four properties matter. First, it is Kraft-tight over binary columns: $\sum_t \binom{n}{t} 2^{-L_{\mathrm{col}}(t,n)} = 1$ to 10^{-15} at $n \in \{1,2,3,5,8,12\}$ (gate UT-1). This property is the premise of the false-mint guarantee. Second, the single-member asymptote is $L_{\mathrm{col}}(1,n) = \frac{3}{2}\log n + \log(2\sqrt{\pi}) = \frac{3}{2}\log n + 1.82575$ (exact against asymptote: 16.7739 against 16.7744 at $n = 10^3$; agreement to four decimals at $n \geq 10^4$). This locates the two-part member $c = \frac{3}{2}$ inside the family $c \log n$, $c \in [1,2]$: 2 for Laplace, 1 for the Bayes-exact prequential price of Lemma 2. Third, the per-member cost obeys $L_{\mathrm{col}}(t,n)/t = H_2(\rho)/\rho + O(\log n/t)$ with $\rho = t/n$ the node's occupancy and H_2 the binary entropy (measured at $n = 10^6$: 2.0000 bits per member at $\rho = \frac{1}{2}$, 4.6901 at $\rho = 0.1$, 8.0803 at $\rho = 0.01$, against $H_2(\rho)/\rho$ of 2.0000, 4.6900, 8.0793). In words: a latent node must carry more than $H_2(\rho)/\rho$ bits per member beyond what the other active nodes and the key-naming code already supply. Nothing in that floor is chosen. Fourth, the incremental form is the attach price $\mathrm{attach}(t,n) = \log \frac{n-t+1/2}{t+1/2}$ (measured: 6.051662 at $t = 1$, $n = 100$; 0.000000 at $t = 50$, $n = 100$; 12.702678 at $t = 1$, $n = 10^4$), negative for $t > n/2$. A node holding more than half the stream is free to join: a derived Matthew effect, not a designed one (gate UT-19). Finally, the exchangeable IBP per-column factor is $\log \binom{n}{m} + \log m$, and L_{col} exceeds it by $\frac{1}{2}\log n + O(1)$, the KT redundancy (Griffiths & Ghahramani, 2011; Broderick et al., 2013b). Two independent code families agreeing on the same $\Theta(\log n)$ shape is the argument; neither alone is.

E3 and E6, the structural bookkeeping. Uniform over supports of a given size is the unique key-permutation-invariant code (Galbrun, 2022); the size header uses the universal integer code. The support charge is material, not decorative: $\log A_n$ is 7.64 bits at $A_n = 200$ and 10.65 at $A_n = 1610$, comparable to a 50-way facet's 5.64. The owner index E6 is the measured bill for the additivity claim. Additivity of the per-facet receipts is bought by single ownership. Single ownership was chosen partly because it yields that additivity. The bill lies in $[0, \log(\text{competing owners})]$ per attribute and is reported as a realised total. We state the circularity rather than smoothing it over.

E4, the most attackable term, and the rule that makes it safe. The rule: a node's presence model covers only its own support. Here is what goes wrong without it. A presence model over the whole key universe hands a fresh node an $|A|$-bit bill (by arithmetic, not an engine run: 1609 bits at $|A| = 1610$, against mint prices of 7.6 to 26.6 bits, a factor of about one hundred), which collapses the graph to one node. Ablation AB-3 is designed to reproduce that failure and is not yet run. With the rule in force the record's total presence bill is $O(m_r + \sum_{v\,\mathrm{active}} |S_v|)$, never $O(|A|)$, because the all-keys-absent background baseline is $\sum_a -\log(1 - \hat{p}_{0,a}) \approx \mathbb{E}[m_r] \cdot 1.4427$ bits, a few bits rather than $|A|$. A 2-key record and a 50-key record are then commensurable, and sparsity cannot decide anything (gates UT-3, UT-8, UT-9). The node-dependence of E4 is dropped nowhere: it cancels from insertion comparisons but is load-bearing in the repair objective (Appendix C.6).

E7, the singleton price. With alphabet d, a block's first observation costs exactly $\log d$. Measured: $L_{\mathrm{KT}}([1,0,\ldots],d) = 1.00000/1.58496/3.32193/5.64386/9.96578$ for $d = 2/3/10/50/1000$, and the normalized maximum-likelihood complexity agrees exactly, $\mathcal{C}(1,d) = d$ to ten decimal places by the Kontkanen-Myllymäki recursion (Kontkanen & Myllymäki, 2007; Shtarkov, 1987). Contrast the plug-in entropy of the same singleton, which is 0.0. That zero is precisely why "insert iff total entropy does not increase" collapses to one node per record. The plug-in is not a code: it assigns probability one to the observed value and violates Kraft. The parametric complexity is exactly the missing term, computed rather than chosen (gate UT-10; ablation AB-4, designed and not yet run, is to reproduce the collapse). One more distinction guards the term: L_{KT} as written is the sequence code, not the count code. Including the multinomial coefficient would make every

block look $NH(1/d)$ bits too cheap, and nothing downstream would complain (gate UT-2). Open alphabets use a three-stage past-only escape in the style of PPM (Cleary & Witten, 1984): a novelty Bernoulli, then an index among seen values, then the key's global inventory, with $\hat{d}_a$ always (distinct values seen so far) plus one.

E8, numeric facets, and the resolution-cancellation lemma. The Student-t predictive carries $\log(1/\delta_a)$, where δ_a is the data's own recorded quantum, inferred over a warmup and frozen. This term appears identically in every candidate owner's cost for the same key of the same record, so it cancels from every decision difference. The condition for the cancellation is that every branch codes the same value at the same resolution, and the condition is load-bearing. The cancellation fails the moment any branch prices a numeric value against a discretised-uniform reference, which would make δ a hidden λ at one extra bit of create price per binary digit of precision. This encoding has no such reference (E10), so the cancellation holds unconditionally, and gate UT-7 is a hard invariance test on exactly this.

E10, the reference. There is no separate fresh-node reference code. In the joint code the alternative to "key a of record r is owned by node v" is ownership by another node or by the background, and the background's block is a real block coded by E7 to E9. The two reference distributions a designer might be tempted to interpolate between already live inside the code. The uniform-over-alphabet pole is what the KT block is at zero counts ($\frac{1/2}{d/2} = 1/d$ exactly); the corpus-predictive pole is what the background block is at large counts. The interpolation weight between them does not exist as a free quantity: its only analogue is the background block's own count state, which is data.

E9, and the honest scope of the claim. There is no minimax code for an unbounded alphabet. The tokenisation residual for free-text facets is estimated at 5 to 15 bits per field, larger than any constant the method removes, so the claim of no calibrated parameter is scoped to categorical and near-categorical facets and text facets are quarantined in v1; ablation AB-12, the criterion with text ablated and included, is designed for the version that lifts the quarantine and is not run.

What is not forced, with its measured price. Three choices remain, and each is priced. First, KT against Laplace on value blocks: 0.05 to 0.92 bits per attribute at sparse counts, decaying like d_a/N; KT ships (Xie & Barron, 2000). On the column the gap is $\frac{1}{2}\log n - 1.826$ bits and grows: 3.16 at $n = 10^3$, 4.82 at 10^4, 8.14 at 10^6. Under the escrow release this is a latency perturbation, not a decision perturbation, and the false-mint guarantee is code-independent; the minted-set agreement at finite n is E12, the encoding-stability sweep (`results/e12_encoding_stability.json`), and it is run. Eleven encodings of three streams, nine of them valid prefix codes, move the description length by 2.3 to 8.0 percent and agree on the graph wherever the graph is determined: on the planted eight-group stream every arm returns the same eight nodes with the same support keys, adjusted Rand index 1.0 between every pair, not one of 3,000 records in a different node, and birth times moving by up to 217 records. Where the shipped code does not itself recover the truth the partition moves as well, to a worst pairwise index of 0.6055 on a shared-key stream and 0.5233 on the Wikipedia infoboxes, against 0.6068 and 0.5804 across twenty arrival orders of the shipped code alone: the code band tracks the order band on the first stream and exceeds it on the second, and 11 of the 13 nodes holding at least five percent of their stream survive the least favourable valid variant. So the claim that the code choice moves when a node is born rather than which nodes exist holds where the structure is identifiable, and not beyond it. Two codes a reviewer might propose are not prefix codes at all: the ceiling form $\lceil \log_2 x \rceil$ has a divergent Kraft sum, and Rissanen's iterated log without its normaliser sums to 2.865. A third item on the list is not a choice either: removing the stage-three naming charge from the value escape sends a block predictive to 85.8 instead of one, a Kraft violation rather than an ablation arm. Second, the two-expert mixtures of E8 and E9 cost at most one bit per key per record. Third, the computational budgets (B_{node} candidates scored per key, B_{cand} escrow candidates tracked, W reservoir slots per node, R repair moves per record) are declared budgets, not modelling parameters; the curve showing decisions unchanged as each grows is designed, not run, and in version 1 only the candidate pool cap B_{cand} exists as a setting, B_{node} is infinite and W and R are not exposed. Shrinking B_{cand} can only reduce the expected number of spurious mints, because the guarantee sums over candidates actually tested; it is a recall budget and never a validity knob.

E THE VERIFICATION SUITE

Terms used here: a gate is a test that ships with the code and must pass for the build to be trusted; L_{col}, L_{KT} and L_{batch} are the column, block and batch codelengths of Appendix D; a range coder is a real arithmetic coder that turns codelengths into bits on disk.

The suite has twenty-two gates. They exist because the failure modes of a codelength criterion are silent: a broken code still returns confident numbers, and four of the six independent derivation routes shipped exactly such a breakage before verification caught it. Gates UT-1 to UT-10 check that the code is a code; UT-11 to UT-22 check that the process built on it behaves as proved. Table 3 and Table 4 state what each asserts and what it catches.

Thirteen of the twenty-two run as executable gates against the v1 implementation (UT-1, UT-2, UT-4, UT-6, UT-8, UT-10, UT-11a, UT-12, UT-13, UT-14, UT-19, UT-20, UT-22, in `code/tests/`); UT-11b is measured by the null experiment of Section 5 on a standalone one-key block, not on the engine. The remaining gates (UT-3, UT-5, UT-7, UT-9, UT-15 to UT-18, UT-21) are specified against the full encoder, including the numeric and text facets and the arithmetic-coder round trip, and are the release gate for the version that lifts the v1 categorical scope. We state this split rather than implying a complete harness. Since that split was written, `code/tests/test_batch.py` (eleven tests, all passing) asserts two further facts the paper relies on: every (record, key) cell is coded exactly once by exactly one block, the invariant that makes the per-key bits additive, and L_{batch} is invariant to relabelling the nodes (difference 0.0 over 20 relabellings); 20 arrival orders that reach the same member partition can still differ by up to 225 bits, because the greedy sequence assigns cell ownership differently.

F REPRODUCIBILITY

Terms used here: a results file is the JSON each experiment script writes and every number in this paper is read from; a seed is the fixed random-number start that makes a run replay exactly; the v1 scope is the categorical-facet scope declared in Section 2.

Repository layout. The implementation is pure Python over the standard library; there is no GPU, no learned component, and no external dependency in the engine. `escrow/codes.py` holds the codelength primitives of Appendix D ($L_{\mathbb{N}}$, L_{col}, L_{supp}, the KT block and predictive forms, the attach price, the mint price, and the Kontkanen-Myllymäki complexity used by UT-10), each computed by `lgamma`, never by factorials. `escrow/engine.py` implements the insertion process of Section 3 (parse, retrieve, per-attribute evidence, greedy active set, emit with receipts, update, escrow accrual, release), with all evidence computed from pre-record snapshots so that the accumulator remains a sequential likelihood ratio. `escrow/batch.py` implements L_{batch} and the repair pass of Appendix C.6. `tests/` holds the executable gates of Appendix E. `experiments/` holds one file per mechanism experiment; each writes a JSON results file that the numbers in this paper are read from.

Declared v1 scope. Categorical facets only; the numeric (E8) and text (E9) coders are specified and quarantined. Posting lists are untruncated ($B_{\mathrm{node}} = \infty$, reported). The repair repertoire is merge over nodes with disjoint member sets, the case the mint produces; the owner-index term E6 is omitted from the shipped batch objective, which is exact for disjoint-support graphs and bounded by $\log(1 + \mathrm{overlap})$ per cell otherwise. Ties in the greedy step break by birth index and then node id, never by hash order, so runs are deterministic given the stream.

Run protocol. Every ESCROW run in the mechanism tier, the demonstrations, E4, E5 and the language-model comparison calls `escrow/protocol.py`: the engine with the batch objective installed, so that a fresh mint whose support is contained in an existing node is absorbed into it, a repair pass every 100 records, and one full pass at the end of the stream; each results file records the protocol string. The three runs on the shared GPU box, Lazada raw keys, MusicBrainz 20K and ReVerb45K, were made on earlier engine states and record their own cadences, stated with each run in Appendix G.

Gate	Asserts	Catches
UT-1	$\sum_t \binom{n}{t} 2^{-L_{\mathrm{col}}(t,n)} = 1$ to 10^{-12}, $n \in \{1, 2, 3, 5, 8, 12\}$	a mis-normalised column code; this identity is the premise of the false-mint guarantee, and if it fails nothing else in the suite matters
UT-2	$\sum_{\mathrm{seq}} 2^{-L_{\mathrm{KT}}} = 1$ over all d^N sequences, $d = 3$, $N \leq 6$	the sequence-versus-count confusion: a multinomial coefficient makes every block $NH(1/d)$ bits too cheap and the method finds structure that does not exist
UT-3	the whole record code sums to 1 over all 27 records of a toy universe, at 0, 1 and 2 nodes	a bad missingness model: charging only present keys sums above one, charging presence twice sums below, and neither shows up anywhere else
UT-4	$\sum 2^{-\mathrm{Price}} \leq 1$ by enumeration at $n \leq 10$, $\lvert A\rvert \leq 4$	a price that has stopped being a code, which voids the false-mint guarantee; the $\log \binom{n}{t}$ leading term is the union bound over which records, and this notices if a refactor drops it
UT-5	per-attribute receipts plus structural terms equal the batch delta to 10^{-9} over 10^4 random pairs, against two independent recomputations	a decomposition whose parts do not sum to the verdict; every per-attribute bit count printed in this paper is downstream of this assertion
UT-6	block-minus-block equals the predictive increment to 10^{-9} over 2×10^4 vectors (measured max error 1.45×10^{-12})	an off-by-one in the escape denominator: monotone, finite, plausible, and wrong by a compounding fraction of a bit
UT-7	halving every numeric quantum changes no decision, no receipt, and the total by exactly one bit per numeric observation	any branch that prices a numeric value against a discretised-uniform reference, which would make the quantum a hidden λ
UT-8	mint decisions uncorrelated with record width ($\lvert\rho_{\mathrm{Spearman}}\rvert < 0.1$); identical key sets with disjoint values must not co-attach, and conversely	an encoding that treats absence as a value and decides everything by sparsity
UT-9	adding 1000 phantom keys changes no decision and shifts the total by exactly the support terms	any per-record $O(\lvert A\rvert)$ presence term, the measured 1609-bit absorption credit that collapses the graph
UT-10	the singleton block costs exactly $\log d$ in two code families; the plug-in entropy 0.0 is never used	the plug-in degeneracy: one node per record, from a quantity that is not a code

Table 3: Code-level gates: the code is a code.

Gate	Asserts	Catches
UT-11	(a) mean final $K \leq 1$ over null streams, run as shipped on 30 streams of 3,000 records where the specification asks 100 of 20,000; (b) empirical $\mathbb{P}(\sup_t G_t \geq b) \leq 2^{-b}$ over 2×10^4 null candidates	an invalid sequential predictive on either side of the ratio, which silently voids the lifetime guarantee
UT-12	two fully disjoint groups give $K = 2$ at every n, with margins growing linearly; the one-key control returns $K = 0$ with a gap flat in n	the hard gate: a criterion that cannot separate two groups sharing nothing invalidates every downstream result; a growing gap at one key means the column code is mispriced
UT-13	on the disjoint fixture the value component of G_t is exactly 0.000 and the presence component about k bits per member; removing the presence term returns $K = 1$	the evidence functional built from value ratios alone, invisible to any test with overlapping keys, and exactly the bug that returned $K = 1$ in all twenty adversarial cells
UT-14	disjoint-support merges are refused and are accepted without the presence guard (E15 recomputes the three pairs at $+289.27/-110.73$, $+2778.26/-421.74$ and $+13822.96/-2177.04$ bits, within 7 bits of the values recorded during development, the presence term contributing exactly $4tk$); the eight-node collapse is a property of the objective's sign, since the shipped repair never proposes a merge for nodes with disjoint supports	the repair pass collapsing the graph because insertion-time non-degeneracy was assumed to transfer to the repair operators
UT-15	L_{batch} identical to 10^{-9} under 200 record permutations	an arrival-position term leaking into the objective, which destroys the termination proof
UT-16	round trip through a real range coder: exact reconstruction, and written bits within 32 of reported bits over 2000 records	every term that is not decodable from the past; nothing else proves the code is a code
UT-17	every counter wrapped: reading any statistic incorporating record n while deciding record n raises, and zero raises occur	the future leak that makes prequential results look better than they are
UT-18	with structure frozen, emitted bits telescope to the batch total within 10^{-6}	double-counted or missing model terms: the two objectives silently diverging is the central claim failing quietly
UT-19	the attach price equals the column increment to 10^{-12} (measured max 8.4×10^{-13}), is 0 at $t = n/2$, negative beyond	an inverted occupancy ratio, which passes every smoke test and produces a graph that avoids popular nodes
UT-20	escape mass and seen mass sum to one; repetition strictly cheapens; unseen values cost at least $\log(u{+}1)$	the standard PPM escape bug, which makes every block slightly too cheap, always in the direction of absorbing everything
UT-21	every accepted repair move strictly decreases L_{batch} recomputed from scratch; the pass ends within $10K$ moves	a delta with the right sign and wrong magnitude, which oscillates forever while each move looks like an improvement
UT-22	identical records give $K = 1$; globally unique records give $K = 0$; a constant key has $\Delta_a = 0.000$ always; a ubiquitous uniform key never justifies an edge	the frequency-versus-lift confusion, which passes the first three controls and fails the fourth

Table 4: Process-level gates: the process behaves as proved.

Gates and experiments, command lines. The gate scripts live under `code/tests/` and the experiment scripts under `code/experiments/`; every command runs from the repository root as `python3 <path>`, with the arguments and seeds shown. Each experiment script writes a JSON results file of the same name (`e7_immediate_vs_deferred.py` writes `e7_immediate_vs_deferred.json`, and so on); the catalogue script, whose arguments name the department and the record cap, runs in fixed order with no seed and writes `catalogue_footwear.json`.

Script	Arguments and seeds	Output
`test_codes.py`	fixed (0)	code-level gates
`test_engine.py`	fixed (0)	process-level gates
`e7_immediate_vs_deferred.py`	0	same-name JSON
`e8_false_mint_null.py`	0; plateau 0 to 11	same-name JSON
`e9_support_curve.py`	0 per cell	same-name JSON
`e13_creation_bias.py`	seeds 11 to 15 per cell	same-name JSON
`e13_tree_arms.py`	five seeds per cell	`e13_tree_arms.json`
`e14_cost_and_scale.py`	0 per cell	`e14_cost_and_scale.json`
`e16_supermartingale_check.py`	fixed	same-name JSON
`e5_order_dependence.py`	seeds 1000 to 1019	same-name JSON
`wikipedia_demo.py`	shuffle 0	same-name JSON
`catalogue_devset.py`	`Footwear 8000`	`catalogue_footwear.json`
`e8_tree_arms.py`	five seeds per cell	`e8_tree_arms.json`
`e4_baseline_army.py`	fixed	`e4_first_tier.json`
`lazada_c2_rawkey.py`	shuffle once	`lazada_c2_rawkey.json`
`mb20k_benchmark.py`	shuffle 0	`mb20k_full2.json`
`e10_reverb45k.py`	fixed	`e10_reverb45k_full.json`

Every seed is a constant in its script, so every run in this paper replays exactly. The Wikipedia run fetches raw infobox templates once and caches them beside the results (`wiki_cache.*.json`), so the exact input records of the reported run are preserved.

What each results file contains. `e7`: on a 6,000-record two-group stream with one percent typos, over five stream seeds, the unseeded immediate mint fires zero times with a mean per-attempt deficit of 31.4 bits (range 30.6 to 32.1), the record-seeded immediate mint under the computed price fires zero times in all twenty cells of a five-alphabet by four-typo-rate sweep, the same rule under a constant price of 16 bits mints only typo-seeded nodes, and the deferred rule ends at $K = 2$ on all five seeds with the two planted supports recovered exactly, identical across seeds, and zero typo-justified nodes (mean 7.4 mint events, range 4 to 9, consolidated by repair). `e8`: the Ville bound measured on a standalone one-key block, not on the engine, over 20,000 null candidates, $\mathbb{P}(\sup_t G_t \geq b)$ of $0.09785/0.0324/0.0064/0.0002/0.0$ against bounds $0.5/0.25/0.0625/0.0039/0.00024$ for $b = 1/2/4/8/12$; and the null plateau, final $K = 0$ on all twelve seeds at every stream length in $\{2, 5, 10, 20\} \times 10^3$. (The VFDT and EFDT comparison arms (Domingos & Hulten, 2000; Manapragada et al., 2018), node counts as a function of stream length at fixed δ, are recorded in `e8_tree_arms.json`, written by `e8_tree_arms.py`; this file records our side.) `e9`: the support curve over the (k, d) grid, $k \in \{2, 3, 4, 6\}$ keys and $d \in \{4, 8, 16\}$ values, cohort size at first mint from $t^* = 11$ at $(2, 4)$ down to $t^* = 3$ at $(6, \cdot)$, monotone in k, with the realised evidence rate and the self-consistent prediction recorded per cell. `e13`: the creation-bias grid answering Kearns et al. (1997), six noise rates by three lengths by three planted counts, five seeds per cell, 270 runs: 10 of the 18 rows converge on K^*, 8 plateau below it, none grows with length, no single run ends above its own K^*, and the breakdown noise is 0.2, 0.3 and 0.4 at $K^* = 2$, 8 and 20. (The planted-stream tree arms, first split index and final node count at both δ settings, are `e13_tree_arms.json`.) `wikipedia`: 320 raw infobox records, 121 distinct raw keys, five nodes under the one protocol on the demonstration order: film 130 (119 film, 9 person, 2 mountain), person 101 (pure), mountain 86 (pure), a second film node of 10 (pure), one politician record isolated; purity 0.9665 over 328 memberships, 8 records in two nodes; the spread over orders is `e5`. `catalogue`: 993 raw footwear records, 43 raw keys, $K = 6$ from 51 mint events, at a measured 303.4 records per second on one CPU core at $K = 6$; the Lazada run recorded 15.49 records per second at $K = 52$ with repair included (`lazada_c2_rawkey.json`, RERUN PENDING). No record-to-record similarity is ever formed. `e14`: the cost-against-K curve and the scale curve, each cell in its own child process so that peak resident memory is that cell's own, with the head-

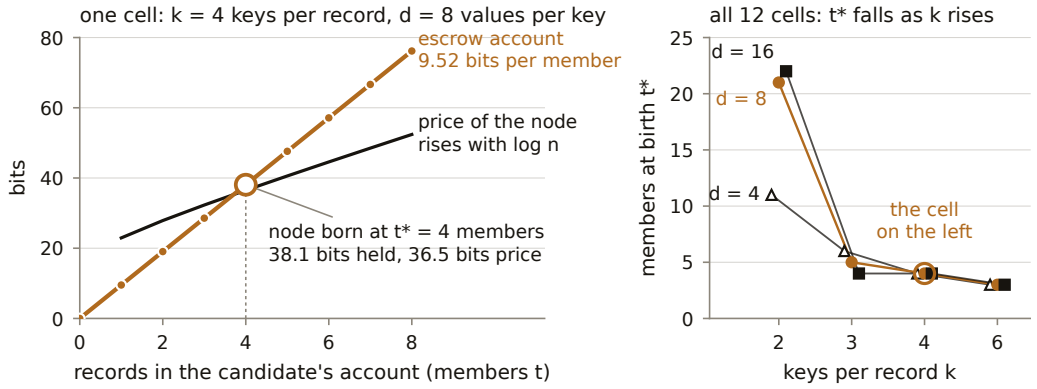


Figure 7: How a node is born. Left: one cell of the two-group fixture, four keys per record and eight values per key, one seed; the account (gold) rises at 9.52 bits per member, the price (ink) with the log of the stream, and the node is born at four members, never before. Right: members at birth across all twelve (k, d) cells fall as k rises, from 11 to 22 at $k = 2$ to 3 at $k = 6$ (results/e9_support_curve.json).

line statistic the median over the last decile of the stream; the insertion step costs 48.58 and 44.98 microseconds per record at 8 and 128 disjoint-support nodes and 113.0 rising to 349.17 at 5 and 20 shared-support nodes, end-to-end throughput falls from $7{,}891.5$ to 908.8 records per second between 10^3 and 10^5 records at a fixed node count, the repair pass takes 95.81 percent of wall time at 10^5, and peak resident memory there is 242.45 MB. `e16`: the martingale audit of Theorem 1, the block-mass algebra checked by brute-force summation on 400 random blocks (maximum error 4.4×10^{-16}), the per-step ratio distribution on four streams, and the exceedance of the accumulator against 2^{-b}.

External benchmarks. The head-to-head tier (Alaska, the AutoPKG protocol on Lazada data (Hongwimol et al., 2026), MusicBrainz 2M and voter records, the DBpedia mappings as the open-domain key-merge test, and the tuned-oracle triples for every knobbed baseline) is designed, with protocols fixed in the planning documents, and is reported as the next tier rather than claimed here; Section 7 states this boundary. The mechanism experiments and controls above are run, and every number in this paper traces to a results file, a findings file, or the code.

F.1 How a node is born, in the two-group fixture

G Additional tables and exhibits

Terms used here: K is the number of nodes and $\hat{K}$ the number the engine ends with; t^* is the cohort size at the first mint and G the escrowed evidence; δ is the significance level of a Hoeffding tree; ARI, F1 and the oracle and transferred conventions are as in Appendix A.

G.1 Results in brief

In brief: on 48 pure-noise streams of up to 20,000 records ESCROW creates no node, where VFDT and EFDT at $\delta = 0.01$ grow to means of 19 and 91 nodes; it recovers two synthetic streams exactly in every one of 20 arrival orders, ARI (the adjusted Rand index, 1.0 a perfect match) of 1.0. On 320 raw Wikipedia infoboxes its agreement with the three category labels depends on arrival order, ARI 0.64 to 0.99 over 20 orders with mean 0.84, the measured cost of a one-pass greedy sequence; a language model on the same records scores 0.99 at about 46,000 tokens per run, and on 1,000 raw product listings returns 24, 78 and 28 nodes across three seeds where ESCROW returns the same six each time. On the free-text benchmark ReVerb45K it is within 0.01 of oracle-tuned embedding baselines on micro F1 with no embedding or knob, but leaves 7,009 of 12,327 subjects as singletons, so its macro F1 is 0.016.

G.2 THE RUN PROTOCOL

Every ESCROW run in the mechanism tier, the demonstrations, E4, E5 and the language-model comparison uses one protocol: the batch objective installed on the engine, a repair pass every 100 records and one full pass at the end of the stream; the three runs on the shared GPU box, Lazada raw keys, MusicBrainz 20K and ReVerb45K, used earlier engine states and the cadences their results files record. An *oracle* knob was chosen on the test labels themselves; a *transferred* knob was tuned on another stream and carried over unchanged, the worse of two transfers reported; k-means *handed the true* K was given the true cluster count.

G.3 E7: IMMEDIATE AGAINST DEFERRED MINTING

The unseeded immediate arm mints zero times in 6,000 attempts, at a mean deficit of 31.4 bits per attempt over five stream seeds (range 30.6 to 32.1): a fresh node's uninformed guess loses to the background on every record, as Lemma 1 says it must. The record-seeded arm under the same computed price also mints zero times, and a sweep over five alphabet sizes $d \in \{2, 4, 8, 32, 100\}$ by four typo rates $\{0, 0.01, 0.05, 0.2\}$ leaves that at zero in all twenty cells. The reason is arithmetic, not luck: the mean cost of a whole typo record under the background is 23.06 bits against a mean node price of 39.79, a typo cell is worth 9.75 bits against a clean cell's 9.33, and even the most expensive single record in the stream, at 33.71 bits, is 6.08 bits short of the price.

That settles which rule the typo objection belongs to. It belongs to the rule the method section describes and rejects, an immediate mint at a *constant* price λ, which E7 also runs on the same stream (59 typo records in 6,000). At $\lambda = 39$ bits it mints nothing. At $\lambda = 16$ it mints 5 nodes and every one of them is seeded by a typo, and none of the five is left as an orphan, so each has gone on to absorb ordinary records and the corruption is now permanent structure with members. At $\lambda = 8$ it mints 200 nodes, of which 30 are typo-seeded and 26 of those are never used again. At $\lambda = 4$ and $\lambda = 2$ it mints 5,998 nodes on 6,000 records, one per record, none ever reused. A constant price therefore has no setting on this stream that both creates structure and refuses typos, which is the failure a computed price does not have.

The deferred arm ends at $K = 2$ on all five stream seeds, with exactly the two planted key sets as supports and the supports identical across seeds, at a mean of 7.4 mint events (range 4 to 9) after repair consolidation, and none of those mints cites a typo value: typos are absorbed, structure is not.

G.4 THE EXPERIMENTAL DESIGN, PER EXPERIMENT

G.5 E8: THE TREE ARMS ON THE NULL STREAMS

The trees do not stay flat because their gate controls error per test and then tests without bound, so no fixed δ holds at every length, and a δ strict enough to silence them buys the silence with delay, since by the Hoeffding bound the records a split needs grow with $\ln(1/\delta)$; ours covers the running maximum of a fair game, so re-testing after every record costs nothing.

The loose δ invents structure on noise. The strict δ is silent here, and $\delta = 10^{-7}$ is river 0.26.1's own default, so the strict setting is the shipped one and the loose setting is the deviation. By the Hoeffding bound the records a split needs grow with $\ln(1/\delta)$, so that silence should be bought with detection delay when structure does arrive. The planted arm measures it (`results/e13_tree_arms.json`, same generator, seeds and δ values as the null arm, so the two tables are two halves of one trade-off).

The delay is real and it is large where the setting matters: on planted structure the strict setting costs VFDT 20,800 records at noise 0, and at noise 0.2 and 0.4 it costs the split entirely. The practitioner must therefore choose δ without knowing which regime the stream is in, and the two regimes want opposite values. EFDT does not pay the delay because it splits at its grace period whatever δ says, but it pays on the other side: it ends with 48 to 311 nodes per fit, cell means 51.4 to 222.6, where eight groups were planted, and the strict setting makes that worse rather than better at both noisy rates. Our own rule re-tests after every record at no cost, because the bound covers the running maximum of a fair game rather than each test separately, and it has no grace period to split at.

G.6 E9: THE SUPPORT CURVE IN FULL

The cohort size at birth falls from 11 to 3 at $d = 4$, 21 to 3 at $d = 8$ and 22 to 3 at $d = 16$.

G.7 E13: THE CREATION-BIAS GRID

The first version of this fixture could not fail: it moved only the value while the key name still named the group, so the key set identified the group at every noise rate and neither failure signature was reachable. The grid below drives value noise (a slot takes a value from another group's alphabet), key noise (a slot takes another group's key name) and a fraction of wholly random records from one dial, so both signatures are reachable, and it is run at five seeds per cell rather than one. Two lengths measured earlier, $T = 30{,}000$ and $T = 100{,}000$, were dropped for runtime: one run at $K^* = 8$ and noise 0.3 costs about 235 s on one core, and the cost concentrates in the middle of the noise range, where a half-broken group manufactures many candidate supports.

Reading the grid. Ten rows converge on K^*, eight plateau below it, none grows with T, and no single run of the 270 anywhere in the grid ends above its own K^*: the largest counts seen are exactly 2, 8 and 20. The steepest slope of mean $\hat{K}$ against T is 0.0021 nodes per record (against $\log T$, 8.71), and the six rows carrying it are all climbing towards K^* from below, which is recovery of planted structure rather than noise absorbed as structure. The breakdown noise, the first rate whose mean ARI at $T = 10{,}000$ falls below 0.5, is 0.2, 0.3 and 0.4 at $K^* = 2$, 8 and 20: it arrives later the more groups are planted, which is what key noise predicts, since with few groups a swapped key lands on one of very few alternatives and wipes the key set out faster. Past it the rule reaches no nodes at all, so the measured failure of this criterion is refusal, not invention.

G.8 E5: ORDER DEPENDENCE

G.9 E4: THE HEAD-TO-HEAD TABLE

Transferred knobs collapse item by item: the agglomerative cut-off is perfect on the two-group stream at 0.05 and 0.0 at 0.1, DP-means on Wikipedia falls from its oracle 0.7286 to 0.0017 with its penalty from either synthetic stream, and the one label-free self-tuner fails likewise, k-means picking two clusters on Wikipedia by the silhouette score for 0.502 against 0.9898 when handed the true count. The two baselines that touch the labels are k-means at 0.9898 handed the true count and agglomerative at 0.9645 with its cut-off chosen on the labels it is scored against.

G.10 CONTROLS AND CODE-LEVEL GATES

Thirteen gate tests ship with the code and all pass (Appendix E states each). Seven sit at the code level and check that the arithmetic is a code at all: the column code and the value-block code each sum to exactly one over their alphabets (Kraft equality within 10^{-12}); the incremental cost update equals the block difference within 10^{-9} over 20,000 random blocks; the price of a singleton block is exactly $\log_2 d$; the escape distribution's total mass is one within 10^{-12} at every step; and the mint price satisfies a Kraft upper bound, so it is a codelength and not a score. The rest sit at the engine level. The *positive control* is exact at every scale tested: $K = 2$ with exactly the planted supports at (k, n) in $(2, 1000)$, $(2, 5000)$, $(4, 200)$, $(4, 1000)$, $(4, 5000)$. The *null control*, 30 iid streams of 3,000 records, passes the $\mathbb{E}[\#\text{spurious}] \leq 1$ gate of Theorem 1. The *degenerate controls* hold: identical records yield at most one node; globally unique values yield $K = 0$; a constant key and a uniformly random key present on every record never justify a mint. The *sparsity control* holds: record widths 2 to 12 from a single population yield at most one node, so mint decisions do not track record width. The *merge guard* holds: on a healthy two-node state the merge delta is strictly positive and the full repair pass keeps $K = 2$.

G.11 RAW RECORDS: A CATALOGUE DEPARTMENT

We stream 993 records of the Footwear department of a raw e-commerce catalogue: keys exactly as published, values kept as opaque strings, no cleaning, no schema. Throughput is 303.4 records per second, with the loop that checks whether a record matches no existing candidate not yet optimised. Over 43 raw keys and 51 mint events the graph settles at $K = 6$, and the nodes are readable.

The largest ($t = 263$ members) is men's casual footwear: `Ideal For` is `Men` on 242 records against `Women` on 29; `Occasion` is `Casual` on 145 against `Sports` on 21 and `Formal` on 10; `Outer Material` splits `Genuine Leather` 64, `Leather` 53, `Synthetic Leather` 31. The second ($t = 58$) is women's heeled sandals: `Type` is `Wedges` 21, `Heels` 18, `Flats` 5, under a `Type` key the men's node does not publish; its `Weight` values are quoted per single sandal (`200 g` on 16 records, `400 g` on 14); its top `Closure` values are `Slip on` 4, `Slipon` 3, `SLIP ON` 3, the same value spelled three ways, surfaced by the node that carries it. Two further nodes isolate records that are not footwear at all: a leather polish (support `Polish Care`, `Applicator Provided`, `Purpose`, `Suitable for`) and incense sticks (`Burning Time`, `Fragrance`, `Number of Sticks per Box`), both misfiled listings whose supports share no key with the footwear nodes. Every mint receipt names its justifying key and its per-facet bits. No label, embedding or similarity function appears anywhere in the run.

G.12 Real-world standing at a glance

G.13 Wikipedia infoboxes: the run in detail

We fetch 320 pages from three unrelated categories: 1994 films (120 records), English male film actors (113), and Alpine four-thousanders, the peaks of the Alps above 4,000 metres (87). From each page we take the raw top-level `key = value` parameters of its first infobox, with no cleaning and no schema, and shuffle everything into one stream of 121 distinct raw keys. Under the protocol of Section 5.1, on the demonstration order, the result is five nodes: a film node of 130 records (119 film, 9 person, 2 mountain), a person node of 101 and a mountain node of 86, both pure, a second film node of 10, pure, and one politician record isolated; purity 0.9665 over 328 memberships, since 8 records belong to two nodes. The film node's discovered support begins `alt_name`, `budget`, `caption`, `cinematography`, `company`, `country`, `director`, `distributor`, `editing`, `editor`, `executive_producer`, `genre`, `gross`, `language`; the person node's is `birth_place`, `children`, `death_place`, `education`, `occupation`, `years_active`; the two mountain nodes share `easiest_route`, `elevation_m`, `first_ascent` and `image_caption`, and separate on `isolation`, `isolation_km`, `listing` and `highest` against `prominence_m`, `range`, `pushpin_map`, `pushpin_relief` and `map_caption`, two infobox templates for mountains (the results file records the first fourteen keys of each support in sorted order); the politician record carries `constituency`, `office`, `party`, `predecessor`, `successor`, `term_start`, `term_end`. This order is one instance: the E4 harness (Table 11) streams the same records under the same protocol on its own shuffle and scores ARI 0.6790 at $K = 5$; the language-model comparison (Appendix G.15) streams them on a third order and scores 0.8744 at $K = 5$; E5 (Table 10) gives the spread over 20 orders, ARI 0.6389 to 0.986 with mean 0.8438 and K from 3 to 6, and the common variation is how the mountains split into nodes. The stream publishes `distributor` on 46 records and `distributors` on one, and the film node's own support holds both `editing` and `editor`: the raw key drift the designed key-identity operator would price.

G.14 Lazada raw keys: the run in detail

The stream is the Lazada marketplace listings released with AutoPKG (Hongwimol et al., 2026): 21,365 listings that carry a specification block, parsed to 3,033 raw keys exactly as sellers published them, shuffled once, no label fed. The graph settles at $K = 52$ nodes covering 8,393 records (39%) at 15.49 records per second, with repair taking 1,180.6 of the 1,384.7 seconds. This run predates the unified protocol of Section 5.1: its results file records a repair pass every 200 records (107 passes) and one at the end, on the batch objective without mint-time absorption, with a fast reassignment override. The largest node (1,510 members) is grocery: `dietary_needs` on 935 records, `packaging_type` on 892, `pack_type` on 888, `organic` on 804, `storage_type_gr` on 685. The second (1,332) is fashion: `clothing_material` on 821, `size` on 721, `color_family` on 700, `fa_pattern` on 697. The third and fourth (1,274 and 1,190) are two placeholder templates on `color_family`, `model` and `brand`, where the top value is `Not Specified` (712 and 957 records). The fifth (694) is supplements and beauty: `units_hb`, `formulation_supplement`, `ingredient`, `skin_type`. The case-variant survey folds keys by lowercasing and stripping every non-alphanumeric character, giving 167 families and 340 pairs of raw keys that differ only in case or spacing. Exactly 1 pair lands in a shared node

(Occasion, 19 records, and occasion, 122, co-supported by the fashion node); 5 pairs have both keys supported somewhere but never together; in 82 pairs one key is supported and the other is not; in 252 pairs neither key is in any support. The file also scores the partition against the silver product type each listing carries in the released knowledge graph, itself model-derived: ARI 0.0008 over 21,260 scored records, with 5,518 silver types against 52 template-level nodes plus one background of 12,906 records. The top nodes do concentrate types (Children's Clothing Set in the fashion node, Nutrient Supplement in the supplements node, Gift Card in a digital-goods node), but concentration is not partition agreement, and the score says so.

G.15 A LANGUAGE MODEL BUILDING THE SAME GRAPH

Terms used here: determinism is the mean pairwise ARI between the three seeded runs of one method, 1.0 meaning the same graph every time; purity is the fraction of records whose node's majority type is their own type; the largest node share is the fraction of scored records in the largest node, the catch-all indicator; a value-named node is one whose name is a value copied from a member record, a value promoted to a node. On Wikipedia every model run formed the same four nodes, Film, Actor, Mountain and Animation, the last a second film node holding one or two records; seeds 1 and 2 each left one record unparsed and placed Animation differently, which is the 0.994. ESCROW formed a film node of 122 records (two mountain records inside), a person node of 112, a mountain node of 74, a second mountain node of 11, and a one-record politician node, the five-node structure of Appendix G.13 on a third arrival order; neither method made a wrong merge. On Lazada the three sampled runs gave $K = 24$, 78 and 28 with pairwise ARI 0.069, 0.375 and 0.051 between them. Seeds 0 and 2 and the greedy run formed department nodes (Toys, Clothing, Electronics) beside a Miscellaneous node holding 403, 392 and 231 of the records and spanning 356, 348 and 217 silver types; seed 1 degenerated: a No_Type node took 851 records spanning 659 silver types, and 60 of its 78 nodes (77 percent) were named after a value copied from one record, such as Model_SYS_CARA7A67/PILVA, 55 of them singletons. Same records, same order, same prompt, and the output changed in kind. The bill was 107,512 (greedy), 109,384, 157,821 and 110,772 tokens per 1,000 records and 1,603 to 1,859 s on one A100; scaled linearly to the 21,365-record stream, a lower bound because the prompt grows with the node list, that is 2.3M to 3.4M tokens and 9.5 to 11 A100-hours. ESCROW's six nodes are spec templates, each with a receipt naming its justifying key and its bits per key: fashion on fa_season (39.6 bits), size, fa_general_styles, fa_pattern and clothing_material; grocery on dietary_needs, organic, pack_type, packaging_type and storage_type_gr; a placeholder template on model (70.4 bits) and color_family; a second placeholder on color_family (122.1 bits), model, brand and Variation; supplements on formulation_supplement (48.2 bits), ingredient and units_hb; books and media on language (73.3 bits), author, isbn_issn, media_format and artist. The run made 24 mints and 18 merges in about 1.5 s; members were identical on all three runs, and 699 of the 1,000 records stayed in the background because their keys carry too little categorical signal for the price (covered-only ARI 0.0301): the model attaches every record it parses, but 23 to 85 percent of them into a catch-all node, which is a wrong merge under another name. Of 35 case-variant raw-key pairs such as color and Color, ESCROW covered both keys of 11 and placed 8 of those 11 in one node; the model placed 13, 29 and 14 of 35 together in the sampled runs and 9 of 35 in the greedy run.

G.16 MUSICBRAINZ 20K, CONDITION A, IN FULL

Condition A hands every method only source, number, year, language and a length band of 30 seconds; title, artist and album are withheld. On the fixed engine ESCROW finds 4,594 nodes at pairwise F1 0.0001 (pairwise recall 0.1703); the run before the engine fixes found 3 nodes at pairwise F1 0.0, with a repair pass every 200 records, a forced pass whenever K exceeded 300 with at least 60 records since the last, and one final pass; one arrival order. Condition B, which adds presence facets for title, artist and album tokens seen at least twice, is pending.

G.17 REVERB45K UNDER CESI'S METRICS, IN FULL

Each record carries its relation phrases as keys, with the sorted objects as the value, plus presence facets for the subject's surface tokens; every baseline renders the same strings to text. ESCROW

ran once, on one arrival order, in 690.1 seconds with 105.9 seconds of repair, minting 75 nodes over 209 records with 12,118 records in the background, on the cadence its results file records: a repair pass every 200 records, a forced pass whenever K exceeded 300 with at least 60 records since the last, and one final pass. The embedding baselines were measured on the same records and protocol in the earlier run and were not recomputed, since they never touch the engine; the file records this. The knobs matter: carried to the worst setting of their own sweeps, HDBSCAN falls to 0.0025, agglomerative to 0.0243 and DP-means to 0.0267 pairwise F1, and single sweep points that go higher on macro alone collapse the other two metrics, HDBSCAN at minimum cluster size 5 reaching macro 0.8915 at micro 0.1140 and pairwise 0.0025. A secondary reading in the file pools the background into one cluster and reports macro 0.5499 at pairwise 0.0025; we do not use it, because one cluster of 12,118 records is not a partition. CESI's own normaliser could not run on the Python on this machine, so the dataset's `triple_norm` field is used for every method, which makes the string space slightly harder than CESI's; the deviation is recorded in the file.

G.18 The remaining external tier

Its protocol is the one the first tier already follows, fixed in advance. Every input our method receives, every baseline receives. Each knobbed baseline is run three ways: oracle-tuned on the test stream, transferred from another stream, and at shipped defaults. ESCROW runs once. The roster, each with its role:

- the anytime-valid betting repair of the Hoeffding gate (Amoukou et al., 2026) on the null streams of E8; the betting form is the same kind of wealth argument our own bound uses;
- the Alaska benchmark as the one-shot key-identity test set against the hand-curated Alaska gold (Crescenzi et al., 2021);
- AutoPKG (Hongwimol et al., 2026), head to head on its own product data under its own evaluation protocol; the opponent there is its decision agent, driven by a large language model; the Lazada run above is the structure exhibit on that data, not the head-to-head;
- the DBpedia mappings wiki, with its thousands of hand-written raw-key-to-ontology mappings, as the open-domain key-identity target at scale;
- MusicBrainz 2M, the full-scale registry behind the 20K benchmark, and voter registries with degenerate nulls, at record level;
- the nested stochastic block model (Peixoto, 2017), a random-graph model with planted communities inside communities, as the hardest structural objection, with HISEvent (Cao et al., 2024) beside it;
- the two adversarial arrival orders of PERCH (Kobren et al., 2017), sorted by gold type and round-robin over types, on the E4 streams, and random orders on the external streams; the random-order spread on the E4 streams is run (E5, Table 10);
- the encoding-stability sweep on which the hidden-knob objection turns is run (E12, Appendix D); its extension to the external streams is what remains designed.

WDC Products, the web product-matching benchmark, was probed and deferred: its identity signal lives in free-text titles, which version 1 quarantines rather than reads.

G.19 Limitations, in detail

Honest scope is part of the claim, so we state it as measurements rather than caveats. **Categorical facets only.** The claim is scoped to discrete facets, and the evaluation matches. Free text is quarantined because any code for it must first choose a tokenisation, worth 5 to 75 bits per field with median 20 over 31 Wikipedia text fields, more than any constant the method removes, and no minimax code exists for an unbounded alphabet. A numeric code is specified, a Student-t predictive whose resolution term cancels from every decision difference, and is quarantined until evaluated. Neither needs new machinery, because the code factorises over facets. **Detection latency.** Deferral is bought with lag. In the two-key cell at eight values the online release fires at record 229, on 21 supporting records; at three keys five records suffice (Table 8). At fifty records the shipped batch objective already prefers the planted two-node structure, by 78.8 bits over the one-node-total code and 69.6 over the all-background code. One caveat belongs beside it: with a single key per group the objective prefers the degenerate code at every stream length, by 12 to 22 bits, which is exactly why the

one-key control returns no nodes; a limit of the criterion, not of the search. On a structureless stream the mint time is infinite, which is the correct answer, not a failure mode. **The search is approximate; the criterion is not.** The exact argmax over candidates is a maximum edge biclique problem and NP-hard (Peeters, 2003), the same wall MDL4BMF names (Miettinen & Vreeken, 2014). The rule evaluates its criterion exactly over a generated pool, one candidate per residual (key, value) pair, under a pool budget that is computational and not part of the criterion. We state that budget as what it is, a declared operational parameter that can change the output, rather than as a formality. When the pool is full the engine evicts the escrow account with the lowest accrued evidence, so the budget decides which accounts survive long enough to be released, and the shipped test measures both endpoints on one stream: at a cap of one, on the shuffled planted stream, each record's fresh candidates evict the previous record's, no escrow ever accumulates and no node is minted at all; on the same records sorted so that a group arrives together, the surviving candidate mints; and at a cap of eight the planted $K^* = 8$ is recovered in full (`code/tests/test_batch.py`, the pool-eviction test). The budget binds on both large real streams, not only in this constructed test: the Lazada run hit the default cap of $4{,}096$ at about record $3{,}600$, where an eviction of a candidate touched by the current record raised a `KeyError` at release, and the MusicBrainz run raised the cap eightfold to $32{,}768$ after the same crash, which removes eviction entirely on that dataset because its whole (key, value) signature space is about $18{,}300$. The engine now never evicts a candidate touched by the current record. What is not run is the curve between those endpoints: a sensitivity sweep over realistic budgets showing the minted set unchanged as the budget grows, which is the report this section owes, since a minted set that moved with the budget would have traded the removed constants for one. The code-choice curve beside it is E12, which is run. Absence evidence costs $O(|\mathcal{A}|)$ work per record in version 1, with $\mathcal{A}$ the set of keys seen so far. **Order dependence.** The batch objective is a function of the state, not of the arrival order: the test suite asserts its invariance to relabelling the nodes (difference 0.0 over 20 relabellings, `code/tests/test_batch.py`). The greedy pass is not order-invariant: which candidate clears escrow first can depend on arrival order, and 20 orders that reach the same member partition can still differ by up to 225 bits because the sequence assigns cell ownership differently. E5 measures the spread of the final structure over 20 arrival permutations (Table 10): none on the two planted streams, and ARI 0.6389 to 0.986 with K from 3 to 6 on Wikipedia. The PERCH adversarial orders (Kobren et al., 2017) are designed, and the one-pass-to-batch gap of Definition C.2, computable on every run with no ground truth, is $9{,}966$ bits and 0.50 per record on the twenty-node stream of Appendix C.6, though the shipped runners do not yet record it per run. **Cost and scale, measured.** The claim that the cost per record does not grow with the node count holds only under a condition, and E14 states it (`results/e14_cost_and_scale.json`; every cell in its own child process, the headline statistic the median over the last decile of the stream, where the graph is at full size). With disjoint supports the insertion step is flat: 48.58 microseconds per record at $K = 8$ and 44.98 at $K = 128$. With every node supporting every key, so that candidate retrieval returns the whole graph, it is not: 113.0 microseconds at 5 nodes and 349.17 at 20, and at planted $K \in \{32, 128\}$ in that regime the repair pass did not finish the stream inside a 180 second budget. Cost per record is flat in stream length, 33.37 microseconds at $n = 3{,}000$ and 35.08 at $n = 10^5$, but the repair pass is not: it takes 22.7 percent of wall time at $n = 10^3$ and 95.81 percent at $n = 10^5$, so end-to-end throughput falls from $7{,}891.5$ to 908.8 records per second over that range at a constant node count. That is the mechanism behind the twentyfold spread between our real-world runs, 303.4 records per second on the footwear catalogue against 15.49 on the Lazada stream. Peak resident memory is 242.45 MB at $n = 10^5$. **Key identity.** A key enters a support only through a minted node; the operator that prices two keys as one under the same code is designed, not run in this version. **Replication.** E4 synthetic and E9 are one seed each, and every real-world run is one arrival order; E5 (20 orders), E7 (five seeds) and E13 (five seeds per cell) report their spread. **Inherited from the price.** The model must not condition on the number of attachments per record; that breaks the complete randomness the martingale rests on, and in the conditioned model the rate's scale cannot be identified even in principle (Doshi-Velez & Williamson, 2015). And $\hat{\gamma}$ sees evidence growing like $\log t$, so after a regime change in the mint rate the price stays miscalibrated for a long stretch. **Scope of the evidence.** The external tier of Appendix G.18 is designed, not run, and on free-text identity version 1 leaves most subjects as singletons (ReVerb45K macro F1 0.016 against micro 0.7777).

G.20 FIGURE SOURCES

Every figure is drawn from a results file by a script under `code/experiments/`; the captions give the headline numbers and the file, and the full data paths are these. Figure 2: the receipt under step 7 is `results/e9_support_curve.json`, cell $k = 4$, $d = 8$, keys `t_star`, `G_at_mint` and `price_at_mint`. Figure 1: `results/llm_graph_formation.json`, keys `datasets.lazada.llm_determinism` (node counts 24, 78, 28; mean pairwise ARI 0.1647), the `largest_node_share`, `tokens_total` and `wall_s` of the four Lazada model runs (0.4038, 0.8527, 0.3928, 0.2315; 109,384, 157,821, 110,772, 107,512; 1,641.5, 1,858.9, 1,629.9, 1,603.2 s), `datasets.lazada.escrow_determinism` and the ESCROW `wall_s` (2.39, 2.41, 2.39 s); column A stands for the agglomerative cut-off and cosine components of E4, column B for DP-means and BP-means, and column C for AutoSchemaKG, AutoPKG and the Qwen2.5-14B run; the cost row for A and B reads not run because those rules were not timed on the Lazada records. Figure 7: `results/e9_support_curve.json`, every cell. Figure 3: `results/e8_false_mint_null.json` (the plateau) and `results/e8_tree_arms.json` (five-seed means; per-seed lists in the file). Figure 4: `results/e4_first_tier.json`, per stream the keys `kmeans_oracleK.ARI` and `kmeans_silhouette.ARI`, and for HDBSCAN, agglomerative, cosine components and DP-means `oracle_ARI` and the minimum over `transferred_ARI`; ESCROW's marks are the per-stream `ARI.mean`, `ARI.min`, `ARI.max` and `K` of `results/e5_order_dependence.json`. Figure 5: node counts after each chunk from `results/llm_gf_runs/` (the four Lazada model runs, `chunks[].records` and `chunks[].nodes_after`); ESCROW's final count from `escrow_lazada_r0.json` to `r2.json`; seed agreement, background share and the Wikipedia scores from `results/llm_graph_formation.json`. No node count over n is stored for ESCROW, so no ESCROW curve is drawn. Figure 8: the same four Lazada model run files (`chunks[].tokens_in`, `tokens_out` and `seconds`), `escrow_lazada_r0.json` (`wall_s`), and `datasets.lazada.total_stream` and `reading.cost` of `results/llm_graph_formation.json`, against which the script checks the extrapolation. Figure 6: the per-record steps are `code/escrow/engine.py` (`process` and `_mint`); the operator order and the full-pass rule are `code/escrow/batch.py` (`repair`); the price is Equation D.10.

G.21 WHAT BUILDING THE GRAPH COSTS

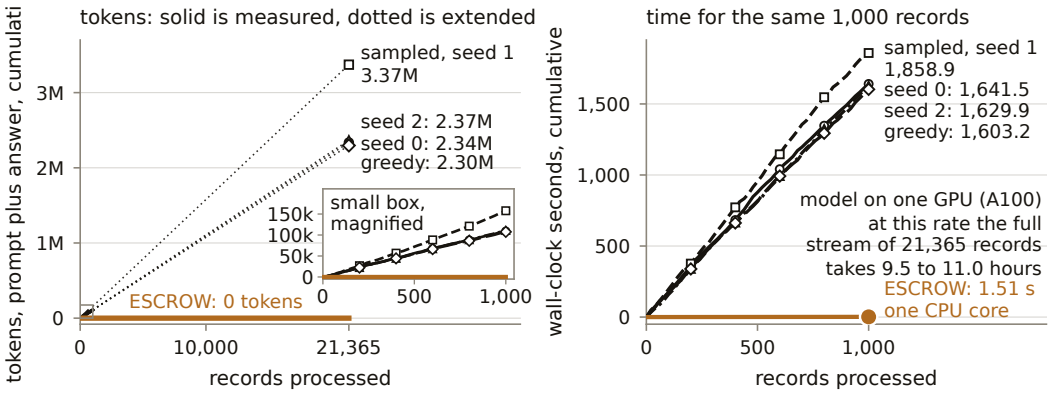


Figure 8: What building the graph costs. Left: cumulative tokens for the four model runs on the first 1,000 Lazada listings, 107,512 to 157,821 (the box at the origin, magnified in the inset), extended at each run's rate to the full stream of 21,365 records, 2.30M to 3.37M tokens, a lower bound. Right: cumulative seconds on one A100, 1,603 to 1,859 s per 1,000 records and 9.5 to 11 hours for the full stream, against about 1.5 s for ESCROW on one CPU core (results/llm_gf_runs/, results/llm_graph_formation.json).

Exp.	Question; design (data, order, what is planted or held out)	Control; baselines and how each was tuned	Metric; falsifier
E7	Can a rule mint the moment a record fails to fit? 6,000 records, two planted groups of three keys, alphabet 8; one percent of records carry a corrupted value that is seen once, ever.	Four arms of the mint rule on the same stream, five stream seeds: unseeded immediate; record-seeded immediate under the computed price, swept over five alphabet sizes by four typo rates; record-seeded immediate under a constant price λ, the DP-means shape, swept over $\lambda \in \{2, 4, 8, 16, 39\}$ bits; deferred, ESCROW as shipped. No external baseline.	Mints, final K, typo-seeded nodes, and how many of those are never reused. Falsifier: an immediate arm that recovers the planted structure without minting on typos.
E8	Does the lifetime false-mint bound hold on a standalone block, and does anything grow on noise? 20,000 null candidate trajectories on one key of alphabet 10, 200 steps, background warmed on 200 draws; null streams of independent uniform draws over four keys, alphabet 10, 12 seeds at each of 2,000, 5,000, 10,000 and 20,000 records.	The null itself: every node created is false. VFDT and EFDT, the Very Fast and Extremely Fast Decision Trees, predicting one key from the other three at $\delta = 0.01$ and at $\delta = 10^{-7}$, which is river 0.26.1's own default, five seeds per cell.	Empirical crossing rate against 2^{-b}; final node count by length. Falsifier: a crossing rate above 2^{-b} at any b, or a node count that grows with the stream.
E9	Does the price behave like a price? Two-group fixture over $k \in \{2, 3, 4, 6\}$ keys per group and $d \in \{4, 8, 16\}$ values per key; the first mint in each cell.	The directional prediction is the control: t^* must fall as k grows. No baseline.	Cohort size at birth t^*; evidence G against price at the mint. Falsifier: t^* not falling with k, or G overshooting the price by more than one member's evidence.
E13	How biased is the creation rule? $K^* \in \{2, 8, 20\}$ planted groups of three keys, alphabet 8; lengths 1,000, 3,000 and 10,000 by six noise rates 0 to 0.5, five seeds per cell, 270 runs. One dial drives three corruptions together: a slot's value drawn from another group's alphabet, a slot's key name replaced by another group's key, and a fraction of records drawn wholly at random and labelled as belonging to no group.	The planted K^*. No baseline; this answers Kearns et al. (1997) by measurement.	Final $\hat{K}$ and ARI against the planted labels. Falsifier: $\hat{K}$ growing with length (noise coded as structure) or plateauing below K^* (structure written off).
E13, trees	What does the strict δ cost when structure is really there? The same planted streams at $K^* = 8$ and noise 0, 0.2, 0.4, with the planted group label as the target, supervision ESCROW never receives; VFDT to $T = 30{,}000$, EFDT to $T = 2{,}000$, five seeds per cell.	The two δ settings against each other on the same records; river 0.26.1's own default is the strict $\delta = 10^{-7}$, and the E8 null arm is the other half of the same trade.	Index of the first split; final node count. Falsifier: a strict δ that costs no delay on planted structure.
E5	How much does record order change the graph? The E4 streams, two-group (2,000), eight-group noisy (3,000) and the 320 Wikipedia infoboxes, each streamed in 20 random orders under the one protocol.	The synthetic streams, where the truth is planted, are the control. No baseline.	ARI and K: mean, standard deviation, minimum and maximum over the 20 orders. Falsifier: K or ARI drifting on planted structure.
E4, Wiki	What does a method with nothing to tune give up, and does the unmodified engine recover types from raw infoboxes? Two-group (2,000 records), eight-group noisy (3,000), raw Wikipedia infoboxes (320 pages from three unrelated categories: 120 films, 113 actors, 87 Alpine peaks; raw first-infobox parameters, shuffled, 121 raw keys); category labels held out.	Five baselines on the same frozen MiniLM sentence embedding, a small pretrained text encoder: k-means (handed the true K, or chosen without labels by the silhouette score, a measure of cluster separation), HDBSCAN, agglomerative, cosine components (records linked when their embeddings are close enough), DP-means, each at oracle and transferred knobs. ESCROW reads raw records, no embedding.	ARI; on Wikipedia also purity per node and K. Falsifier: baselines keeping their oracle scores under transfer, or ESCROW trailing an honestly run baseline; mixed nodes, or a node count unrelated to the categories.
Lazada	Does it find the spec templates a marketplace is built from, and does it group case-drifted keys? 21,365 spec-bearing listings, 3,033 raw keys as published, shuffled once, no labels.	A structure exhibit by protocol, no embedding baseline; a case-variant survey of 340 raw-key pairs; the released silver product type (a model-derived label) as a sanity score.	K, coverage, pairs in a shared node. Falsifier: K in the hundreds or unreadable supports; or drifted keys landing in shared supports as a matter of course.
MB20KA	What happens where the visible fields carry no identity signal? 19,375 songs, five categorical fields (source, number, year, language, length band), 10,000 true entities; title, artist and album withheld.	The null-signal input. k-means handed $K = 10{,}000$; HDBSCAN, agglomerative, DP-means at oracle knobs on the same embedding.	Pairwise F1, ARI, K. Falsifier: a node count that grows to fill the record count.
ReVerb	Can relation and token facets alone recover entity identity? 12,327 subject strings, 36,814 mentions, 6,061 gold clusters, CESI's granularity and metrics code.	The no-merge floor: same-string mentions are one cluster for every method by construction. k-means handed 6,061; HDBSCAN, agglomerative, DP-means oracle; CESI's published numbers as context, with side information no method here sees.	Macro, micro, pairwise F1 (averaged over clusters, over records, over record pairs); pairwise precision. Falsifier: not a low score, but merges that pull pairwise precision below the no-merge floor, meaning they cross entity lines.
LLM	Does a language model that builds the graph directly form it well at demo scale and at production scale? The 320 Wikipedia infoboxes (3 gold types, 121 raw keys) and the first 1,000 listings of the Lazada stream (643 raw keys), each in one fixed arrival order shared by both methods; the Lazada silver truth is AutoPKG's model-derived product type, 759 types over 1,000 records, 614 of them singletons.	ESCROW on the identical records and order, three repetitions, under the one protocol of Section 5.1. Qwen2.5-14B-Instruct (bf16, one A100) prompted to create or attach: chunks of 20, the current node list in the prompt, a JSON answer per record, no cluster count given. Not tuned: sampled at temperature 0.7 with seeds 0, 1, 2, and once greedy.	ARI, purity, K; determinism (mean pairwise ARI across seeds), largest node share, value-named nodes, tokens, wall time. Falsifier: a model that is deterministic across seeds and forms no catch-all node at scale.

Table 5: Experimental design, one row per experiment. VFDT and EFDT are Domingos & Hulten (2000) and Manapragada et al. (2018); HDBSCAN is hierarchical density-based clustering; DP-

	δ	T=2,000	T=5,000	T=10,000	T=20,000
VFDT	0.01	1.0	1.0	1.0	19.0
EFDT	0.01	15.0	73.0	73.0	91.0
VFDT	10^{-7}	1.0	1.0	1.0	1.0
EFDT	10^{-7}	1.0	1.0	1.0	1.0
ESCROW	none	0	0	0	0

Table 6: E8, tree arms on the iid null streams: mean final node count over five seeds per cell; the per-seed EFDT values at $\delta = 0.01$ and $T = 20{,}000$ are 111, 111, 111, 11 and 111, so the maximum is 111. The two scales differ by one because a silent tree still has its root: tree silence reads 1, and our silence reads 0 minted nodes. Every tree node beyond the root is false structure. Our row is the 48-of-48 result of `results/e8_false_mint_null.json` (12 seeds per length).

		first split, record index		final nodes	
	noise	$\delta = 0.01$	$\delta = 10^{-7}$	$\delta = 0.01$	$\delta = 10^{-7}$
VFDT, $T = 30{,}000$	0.0	8,400	29,200	20.2	9.0
	0.2	9,400	never	129.0	1.0
	0.4	8,800	never	129.0	1.0
EFDT, $T = 2{,}000$	0.0	200	200	51.4	51.4
	0.2	200	200	101.4	177.6
	0.4	200	200	131.8	222.6

Table 7: The planted arm of the tree comparison, $K^* = 8$ planted groups, the planted label as the target, five seeds per cell; means over seeds. *Never* means no split inside the stream on any of the five seeds. VFDT's first split at noise 0 is record 8,400 on all five seeds at the loose setting and 29,200 on all five at the strict one, so the strict setting costs 20,800 records of delay; at noise 0.2 and 0.4 it never splits at all inside 30,000 records where the loose setting splits at 9,400 and 8,800. EFDT's setting buys nothing on this stream: it splits at record 200, its grace period, in all thirty fits at both settings. VFDT runs to $T = 30{,}000$ and EFDT to $T = 2{,}000$ because one EFDT fit costs about 79 s at $T = 2{,}000$ and did not finish inside twenty five minutes at $T = 10{,}000$ on one core.

k	d	t^*	n at mint	G at mint	price (bits)	bits/member
2	4	11	46	55.458	54.775	5.042
2	8	21	229	122.987	122.960	5.857
2	16	22	548	158.305	156.739	7.196
3	4	6	20	40.497	37.129	6.750
3	8	5	33	41.177	40.807	8.235
3	16	4	23	36.389	35.131	9.097
4	4	4	12	32.924	32.033	8.231
4	8	4	20	38.085	36.505	9.521
4	16	4	41	47.629	42.489	11.907
6	4	3	8	37.854	32.972	12.618
6	8	3	10	43.035	34.594	14.345
6	16	3	12	38.849	35.885	12.950

Table 8: E9, the support curve: cohort size at birth t^*, stream position n, accumulated evidence G and price at the first mint, per (k, d) cell; k is keys per planted group and d the alphabet size; one seed per cell. t^* falls as k grows, and the overshoot G minus price is below one member's average evidence (last column) in every cell. Across d both competing effects are visible: at $k = 2$ thinner values raise the price and t^* rises (11, 21, 22); at $k = 3$ they make the background's predictions worse and t^* falls (6, 5, 4); at $k \geq 4$ the two cancel.

		T=1,000		T=3,000		T=10,000		
K^*	noise	$\hat{K}$	ARI	$\hat{K}$	ARI	$\hat{K}$	ARI	regime
2	0.0	2.0	1.000	2.0	1.000	2.0	1.000	converges
2	0.1	1.0	0.530	1.0	0.547	1.0	0.553	plateau below
2	0.2	0.0	0.000	0.0	0.000	0.0	0.000	plateau below
2	0.3	0.0	0.000	0.0	0.000	0.0	0.000	plateau below
2	0.4	0.0	0.000	0.0	0.000	0.0	0.000	plateau below
2	0.5	0.0	0.000	0.0	0.000	0.0	0.000	plateau below
8	0.0	8.0	1.000	8.0	1.000	8.0	1.000	converges
8	0.1	8.0	0.753	8.0	0.854	8.0	0.895	converges
8	0.2	8.0	0.457	8.0	0.563	8.0	0.621	converges
8	0.3	2.0	0.081	7.4	0.397	8.0	0.443	converges
8	0.4	0.6	0.012	1.0	0.019	2.8	0.017	plateau below
8	0.5	0.0	0.000	0.0	0.000	0.0	0.000	plateau below
20	0.0	20.0	1.000	20.0	1.000	20.0	1.000	converges
20	0.1	20.0	0.674	20.0	0.827	20.0	0.893	converges
20	0.2	15.6	0.283	20.0	0.496	20.0	0.679	converges
20	0.3	3.6	0.038	19.6	0.409	20.0	0.504	converges
20	0.4	0.0	0.000	8.0	0.130	20.0	0.461	converges
20	0.5	0.0	0.000	0.2	0.004	1.8	0.025	plateau below

Table 9: E13, creation bias: mean final $\hat{K}$ and mean ARI against the planted labels over five seeds, for three planted counts by six noise rates by three stream lengths, 270 runs. The regime column applies the rule declared in the results file: *converges* is mean $\hat{K}$ within 0.5 of K^* at the longest stream, *plateau below* is mean $\hat{K}$ below $K^* - 0.5$ there; no row meets the growing-with-T rule (slope on $\log T$ at least 0.5 and mean $\hat{K}$ above $K^* + 1$), and no row sits above K^* at all.

stream	n	K^*	ARI mean	sd	min	max	K: mean (min to max)
two-group	2,000	2	1.0	0.0	1.0	1.0	2 (2 to 2)
eight-group noisy	3,000	8	1.0	0.0	1.0	1.0	8 (8 to 8)
Wikipedia infoboxes	320	3	0.8438	0.1029	0.6389	0.986	4, sd 0.77 (3 to 6)

Table 10: E5, order dependence: the same records streamed in 20 random orders under the one protocol (seeds 1000 to 1019). Planted structure is recovered exactly on every order; on the real stream the greedy one-pass sequence is order-dependent and the spread is its measured cost; this spread is ESCROW's Wikipedia result. The 0.6790 of E4's order (Table 11) and the 0.8744 of the demonstration order (Table 13) are two instances of it.

method	two-group (n=2,000, K^*=2) oracle / transferred	eight-group noisy (n=3,000, K^*=8) oracle / transferred	Wikipedia infoboxes (n=320, K^*=3) oracle / transferred
k-means (handed K / silhouette)	1.0 / 1.0	0.9977 / 0.9977	0.9898 / 0.502
HDBSCAN	1.0 / 0.5121	0.9565 / 0.0184	0.4989 / 0.0
agglomerative	1.0 / 0.0	1.0 / 0.0	0.9645 / 0.0002
cosine components	0.0 / 0.0	0.0 / 0.0	0.4938 / 0.0002
DP-means	0.994 / 0.0	0.9845 / 0.0	0.7286 / 0.0017
ESCROW (ours), 20 orders	1.0, K=2	1.0, K=8	0.8438, 0.64 to 0.99, K 3 to 6

Table 11: E4, head to head: ARI against the true labels, as oracle / worst transferred. Oracle tunes the knob on the test labels; transferred tunes it on another stream and reports the worse of the two transfers. k-means has no knob: its first value is handed the true cluster count, its second picks the count by the label-free silhouette score. All baselines run on the same frozen MiniLM embedding, each a single run on one fixed embedding; ESCROW runs on the raw records with no embedding and no calibrated parameter, and its row is the mean and range over the 20 arrival orders of Table 10 (the synthetic entries hold on all 20), of which E4's own order gives 0.6790 at $K = 5$.

stream (records)	ESCROW, run once	strongest comparison without labels	strongest comparison touching labels
Wikipedia infoboxes (320)	over 20 orders ARI 0.8438 mean, 0.6389 to 0.986, K 3 to 6 (E5). On the demonstration order $K = 5$: film 130 (119 film, 9 person, 2 mountain), person 101, mountain 86, a second film node of 10, one politician; purity 0.9665	k-means with silhouette-chosen $K = 2$: ARI 0.502; every transferred knob scores 0.0 to 0.0017	k-means handed $K = 3$: 0.9898; agglomerative at oracle threshold 0.7: 0.9645
Lazada raw keys (21,365)	$K = 52$ covering 8,393 records (39%); 1 of 340 case-variant key pairs in a shared node, 252 with neither key in any support	none by protocol: a structure exhibit	silver product type, itself model-derived: ARI 0.0008 against 5,518 types
MusicBrainz 20K, condition A (19,375)	$K = 4{,}594$; pairwise F1 0.0001, ARI about zero	none run without labels	k-means handed $K = 10{,}000$: 0.0016; HDBSCAN oracle 0.0075; agglomerative oracle 0.0003; DP-means oracle 0.0005
ReVerb45K test (12,327)	75 nodes over 209 records, 12,118 in the background; macro F1 0.0160, micro 0.7777, pairwise 0.7160 at pairwise precision 0.7690	the no-merge floor, any method that merges nothing: pairwise 0.7677 at precision 0.9035	DP-means oracle: micro 0.7871, pairwise 0.7708; k-means handed $K = 6{,}061$: macro 0.3537; CESI as published, with side information: macro 0.627, micro 0.844, pairwise 0.819
Wikipedia (320) and Lazada first 1,000, against a language model (Table 13)	Wikipedia: ARI 0.8744, purity 0.9656, $K = 5$, identical on three runs, 0 tokens, 0.17 s. Lazada: $K = 6$ spec templates, identical on three runs, 699 of 1,000 records in the background, 0 tokens, 1.5 s	Qwen2.5-14B-Instruct building the graph by prompting, no labels fed: Wikipedia ARI 0.9905, purity 1.0, $K = 4$, determinism 0.994, about 46,400 tokens and 365 s per run; Lazada determinism 0.1647, K of 24, 78 and 28, largest node 0.5498 of records, 107,512 to 157,821 tokens per run	the Lazada silver product type on the same 1,000 records, 759 types: ARI 0.0002 (ESCROW), 0.002 (model, sampled mean), 0.0094 (model, greedy)

Table 12: Real-world standing at a glance. Every number is copied from the results file named in the source comment; the runs are detailed in the subsections that follow, and each is one arrival order. A transferred knob is the worse of two transfers; an oracle knob is chosen on the test labels.

stream	method	ARI	purity	K	determinism	largest node	tokens	wall (s)
Wikipedia (320 records, 3 gold types)	model, sampled, mean of 3 seeds	0.9905	1.0	4	0.994	0.3709	46,384	364.5
	model, greedy	0.9896	1.0	4	run once	0.3688	46,420	353.2
	ESCROW, three runs	0.8744	0.9656	5	1.0	0.4062	0	0.17
Lazada, first 1,000 (759 silver types)	model, sampled, mean of 3 seeds	0.002	0.0778	43.3	0.1647	0.5498	125,992	1,710.1
	model, greedy	0.0094	0.0802	28	run once	0.2315	107,512	1,603.2
	ESCROW, three runs	0.0002	0.0371	6	1.0	0.0701	0	2.40

Table 13: A language model building the same graph, against ESCROW on the identical records in the identical order. The model is Qwen2.5-14B-Instruct (Qwen Team et al., 2024) in bf16 on one A100, prompted to create or attach: chunks of 20 records in arrival order, the current node list in the prompt, a JSON answer per record naming an existing node or a new one, no cluster count given; sampled at temperature 0.7 with seeds 0, 1, 2 (the mean is reported; the sampled-mean K of 43.3 is 24, 78 and 28) and once greedy, which ran once by design. ESCROW ran three times on one arrival order under the one protocol of Section 5.1. ESCROW's background (745 of the 1,000 Lazada records, none on Wikipedia) is not a node and is not counted in its largest node share. The Lazada silver truth is AutoPKG's model-derived product type, 759 types over 1,000 records, 614 of them singletons, so ARI against it is near zero for every method. Tokens are prompt plus completion under the model's own tokenizer; wall time is per run, the model on one A100 and ESCROW on one CPU core including repair. Every number is from `results/llm_graph_formation.json`.

method	pairwise F1	clusters
ESCROW (ours), run once	0.0001	4,594 nodes
k-means, handed K=10,000	0.0016	10,000
HDBSCAN, oracle min size 3	0.0075	7,327
agglomerative, oracle threshold 0.1	0.0003	1,924
DP-means, oracle λ=0.1	0.0005	1,990

Table 14: MusicBrainz 20K, condition A: five categorical fields, no text, one arrival order. Pairwise F1 against the 10,000 true entities over 19,375 records. Every method is null. ESCROW mints 4,594 nodes at pairwise F1 0.0001; each baseline emits thousands of clusters of noise. Baseline knobs are chosen on the test labels.

method	macro	micro	pair	pair prec.	clusters
ESCROW (ours), run once	0.0160	0.7777	0.7160	0.7690	12,192
HDBSCAN, oracle min size 50, no merges	0.0138	0.7830	0.7677	0.9035	12,327
agglomerative at 0.1, best pairwise F1	0.0391	0.7849	0.7679	0.9036	12,203
agglomerative at 0.2, best macro F1	0.1002	0.7863	0.7192	0.7287	11,822
DP-means, oracle λ=0.25	0.0504	0.7871	0.7708	0.9034	12,142
k-means, handed K=6,061	0.3537	0.6559	0.3729	0.2388	6,061
CESI, published, side information	0.627	0.844	0.819		

Table 15: ReVerb45K test set under CESI's metrics code, one arrival order: macro, micro and pairwise F1, pairwise precision, and cluster count over 12,327 subject strings (6,061 gold clusters), background records counted as singletons. A method that merges nothing scores pairwise 0.7677 at precision 0.9035 because same-string mentions are one cluster by construction; ESCROW's 75 nodes lower precision to 0.7690, so some merges still cross entity lines, and 7,009 subjects end as singletons, which is why macro F1 is 0.016. Only k-means, handed the gold count, exceeds macro 0.35. CESI's published row uses side information no method here sees.