Certificates, not printouts: exact-rational bounds for probabilistic programs

a same-box comparison against GuBPI and Diabolo on their own test cases, preregistered, with the losing rows left in the table

davidad's AI ensemble*
MAITRIA — Mathematical Assurance and Interaction Toolkit for Reliable Intelligent Agents

2026-08-03 — draft for discussion

1 Scope, sources, and how claims are graded

Two published tools compute *guaranteed* bounds on the posterior distributions of probabilistic programs: **GuBPI** (Beutner, Ong, and Zaiser, PLDI 2022 [1]) and **Diabolo** (Zaiser, Murawski, and Ong, POPL 2025 [2]). Both are sound by careful semantic argument; neither emits a replayable witness, and neither has an independent checker — the number printed at the end of a run is the run's own testimony. This report describes a two-day experimental campaign (2026-08-03/04) that ran nine benchmark programs drawn from the two tools' own suites through the maitria toolchain — an untrusted producer emitting exact-rational certificates, checked independently by a small engine that re-derives every number from stored rows — and compared the results against both baselines on the baselines' two native axes, bound tightness and wall-clock, on the same machine, under preregistered claims.

The headline, stated up front with its qualifications attached: on every case either baseline shares with us, the certified bracket wins or ties against that baseline at matched configuration — every tie at an exact point or at print precision, the strict wins reaching $9.2 \times 10^7\times$, and no tightness loss standing (the first edition's one near-miss was converted to an exact width-zero point by a rule revision, §5.4, whose second receipt is the one case both baselines run); the end-to-end wall — *including* certificate checking — beats every baseline that runs the case on six of the nine cases (2.2× to 411×, including 38× on the baseline suite's slowest solved row and both baselines at once on the case both run), and beats GuBPI by up to 713× on the shared trio; and every number ships as a replayable certificate with a mutant battery, an axis neither baseline occupies. The report also contains the rows where we lose: preregistered wall-clock marks missed on two cases (one by a factor of thirty), a flagship cell over its own preregistered wall, and a lane where Diabolo's millisecond-class residual runs outrun our current checking path outright. §8.5 collects these; none is hidden in a footnote.

Pinned artifacts. The comparison is pinned:

- GuBPI at commit `47e9a21` (2023-04-21, repository head at campaign time; MIT), built from source on the measurement box, with its bundled VINCI polytope-volume tool.
- Diabolo at commit `6e2c3e0` (2024-12-05; MIT), built from source on the measurement box. Benchmark program files were additionally diffed byte-for-byte against the POPL 2025 Zenodo artifact (DOI `10.5281/zenodo.14169507`); the artifact's README supplied the published comparison commands (§6.4).
- The maitria side at the `probprog/` experiment family of the `qtsl-experiments` repository (main branch at or after `05f91e7`), whose make targets regenerate every number in this report (§9); the certificate species and their machine-checked soundness proofs live in the `qtsl` repository.

*Drafted primarily on Claude Fable 5; the baseline builds, same-box measurements, mutant batteries, and Lean mechanizations were produced by the same ensemble, and the adversarial verification steps (differential oracles, mutant runs, preregistration scoring) are recorded in the reproduction receipts beside the numbers. The authors built the toolkit being compared; §1 states the resulting discipline.

Claim grades. Three grades are used, per the maitria claim ledger. [**shipped**] means a fresh checkout reproduces the number from its make target with the certificate checked against a sealed theory revision. [**argued**] means the receipts exist and the checks are green in the workshop checkout, but at least one pending step separates the claim from [**shipped**] (in this report's first edition that step was the certificate species being *staged* rather than sealed; in this edition it is the 3-D row's pattern face awaiting its own revision event). [**aspirational**] marks design intent with no artifact yet. **Every absorption-bracket row and every finite-face expected-value row in this report is [shipped] as of the specification theory's revision event that this report's first edition described in the future tense.** That event executed; its record (§9.1) discharged the first edition's own preregistered requirements: byte-identical re-emission of every container, both mutant batteries in class through the flag-free checking lane, the development-oracle lane retired for the finite face, and the staged-lane pin collapsed into the repository's standard theory pin. **The single remaining [argued] scope is the 3-D expected-value row**: its pattern face was deliberately deferred at the event's adversarial review (§9.1), and its verdicts stay on the development oracle, labeled, until that face's own event.

Bias disclosure. The authors built the toolkit being compared, and a benchmark report written by the home team has a predictable failure mode: favorable configurations, quiet omissions, post-hoc claim selection. Five disciplines are applied against it, and the reader should audit all five rather than trusting any. (1) *Preregistration*: per-case claims, expected verdicts, mutant loci, and wall-clock targets were committed to the repository before the corresponding runs; deviations are reported as deviations (§6.5), including one preregistered derivation that the campaign's own oracles falsified. (2) *Baselines rebuilt and validated first*: both tools were built from source and their published results reproduced on our hardware before any comparison was made; where our reproduction disagreed with a published table, the discrepancy was chased to its documented root before proceeding (§6.4). (3) *Baselines at their own configurations*: every baseline run uses the configuration embedded in the baseline's own artifact — its flag headers, its comparison scripts, its Zenodo README commands — never a configuration we chose. (4) *Same box, both sides, end to end*: all comparative walls are process end-to-end medians on one named machine, with the maitria side charged for certificate checking; the baselines' internal self-timings and their papers' published numbers (hardware unstated in both) appear as context only, never as the comparator. (5) *Non-wins are rows*: cases and axes where a baseline wins are reported in the same tables as the wins, and collected in §8.5. Residual bias should be assumed anyway.

One register note: this report compares *artifacts*, not research programs. GuBPI and Diabolo are serious, sound tools whose benchmark suites are — deliberately — hard targets; the campaign's most useful outputs were the places their cases forced the maitria specification calculus to grow (§5), and the generous reading of both tools in §3 is the intended one.

Structure. §2 states the architecture and §3 reads the two baselines. §4 explains the method itself — how a probabilistic program becomes a Markov-kernel hypernet, how exact-rational chain certificates bound its posterior and its moments, what the producer actually searches, and the exact support boundaries for distributions (§4.2), loops (§4.5), and conditioning (§4.6); background on the semantic universe and the specification calculus is deferred by citation to the MAITRIA book [5]. §5 gives the certificate species the benchmarks forced, §6 the protocol and audit apparatus, §7–9 the cases, results, and reproduction spine.

2 The thesis instantiated

The maitria toolkit's organizing principle is **powerful untrusted producers, small trustworthy checkers**: any machinery at all — floating-point optimizers, GPUs, learned search, a language model writing programs that guide the search — may *propose* a claim, and being wrong costs a rejected certificate, never a false verdict, because the only trusted component is a small checker that re-derives the claim from explicit certificate content. This campaign is that principle instantiated on the guaranteed-bounds problem for probabilistic programs:

- **A bound is a checked artifact, not a tool output.** The producer emits a certificate: a container of rows — the chain's stored transition rows, class labelings, horizon, and claimed brackets — in the maitria specification theory. The checker (the geolog fast-path engine with the QTSL plugin) recomputes every mass from the stored rows by exact rational arithmetic and accepts or refuses. Derived quantities are never stored, so forging an intermediate value is unrepresentable rather than detected.

- **The producer is untrusted by construction.** The producer used here is a small Rust program (exact integer-scaled dynamic programming over layered chain encodings, plus template search for contraction certificates); nothing in the trust story depends on it. Its unrolling depths and template choices are its own search freedom, exactly as interior-point settings are Diabolo's — disclosed in the receipts and priced into our wall-clock, never into the trust base.
- **The comparison is a forcing function, not the artifact.** The point of running the baselines' own cases was to push the specification calculus toward greater generality: a case an existing certificate rule covers is validation; a case that demands a new or strengthened rule is the high-value outcome, and the campaign produced both kinds (§5). None of the four certificate-row families this experiment class needs existed in any prior revision of the specification theory — the benchmarks genuinely forced the growth.

The trusted-kernel framing itself is not new — it is the classical architecture of proof-carrying artifacts, and its tension with Bayesian-inference tooling (inference engines are large, clever, and unauditable exactly where their outputs matter) has been noted in the probabilistic-programming community. What this campaign adds is an existence proof at benchmark speed: the checked path is not a 10× tax on the unchecked one; on most of these cases it is faster than the incumbent unchecked tools by two to three orders of magnitude while producing tighter bounds and a witness.

3 The baselines, read generously

GuBPI. GuBPI [1] computes guaranteed bounds for a *universal* probabilistic programming language: recursion, higher-order functions, continuous sampling, soft conditioning via `score`. Its method — interval-trace semantics with trace-space splitting, validated-float interval arithmetic, polytope volumes for the continuous fragment — covers a program class far beyond what this campaign certifies (our species is discrete-chain content; GuBPI's continuous and recursive suites are future targets, not present ones, and this report claims nothing about them). On the discrete cases compared here it is the slower and looser tool, which its own authors documented first: the Diabolo paper's comparison table is built on exactly that observation. Its arithmetic is validated-float where it matters, with one documented exception we inherit as a caveat (its bundled volume tool's output is trusted unvalidated), and one benchmark defect its artifact README states plainly (it does not successfully normalize the die-paradox case; the comparison below follows the artifact's own practice for that row). The suite also contains cases where GuBPI *refuted* published bounds of an earlier tool — the reason guaranteed bounds are worth having, demonstrated by the baseline itself.

Diabolo. Diabolo [2] computes guaranteed bounds for discrete $\mathbb{N}$-valued programs with unbounded loops and hard conditioning, via two semantics: residual-mass (exact finite unrolling plus a rigorous remainder) and geometric-bound (contraction invariants found by numerical optimization, then verified exactly inside the tool — a float-propose/exact-verify architecture that is philosophically half of our own thesis, executed within one process). It is *fast*: its residual-mass rows run in milliseconds, and this campaign's one standing wall-clock non-win (§8.5) is against exactly that lane. Its exponential tail bounds $O(c^n)$ automate something no earlier tool did. Its arithmetic is exact rational throughout the verification path. What it lacks — by scope, not by defect — is any artifact a third party can re-check without re-running the tool: no witness format, no external checker, and its verified invariants die with the process that found them.

Both tools are MIT-licensed, which is what allowed their benchmark programs to be vendored verbatim into our reproduction spine and their binaries rebuilt on our hardware. Both papers omit hardware specifications for their published timings, which is why the same-box protocol below re-measures everything and treats published numbers as context. And both suites did exactly what benchmark suites are for: every certificate species described in the next section exists because one of their cases demanded it.

4 Method: from a program to a checked bound

Everything in this report rests on two moves. First, a probabilistic program denotes a *Markov kernel presented by a deterministic dataflow graph* — a hypernet — with randomness confined to explicit noise ports, so the kernel arrives as exact-rational data rather than as a measure-theoretic abstraction. Second, a *bound* on such a program is a set of claim rows about the induced chain — brackets on eventual-absorption measures, posteriors, moments, tails — that a small checker re-derives from stored transition rows by exact

rational arithmetic, per the organizing sentence of §2: the producer that finds the rows may be arbitrarily clever and is completely untrusted; only the re-derivation is trusted. This section explains both moves, states what the producer actually searches (§4.4), and draws the exact support boundaries — which distributions (§4.2), which loop shapes (§4.5), which conditioning primitives (§4.6). Background on the semantic universe and the specification calculus in which these sentences are formal is deferred to the MAITRIA book [5] (the chapters *The Semantic Universe*, *Hypernets: Primitive Functions*, and *Hypernets: Markov Kernels*); the bounding mechanism itself is self-contained here, and the resulting artifact is dissected byte by byte in §6.2.

4.1 Programs as Markov-kernel hypernets

The toolkit's base calculus is deterministic: a *hypernet* is a dataflow graph over a fixed roster of ten primitive boxes on exact-rational data, with one evaluator, one canonical serialization, and one content digest. A *Markov-kernel hypernet* is an ordinary hypernet together with a designated roster of *noise ports*: inputs read as fresh uniform draws on $[0, 1]$. Its denotation is the Markov kernel obtained by pushing the uniform reference measure forward through the deterministic semantics — every kernel is presented by an explicit deterministic dilation with its randomness exposed at the interface, never hidden inside a box. Three consequences come for free, and all three are machine-checked in the toolkit's Lean development at the finite-outcome stratum [5]: *normalization is a theorem, not a constraint* (every box is total and the reference measure is a probability, so nothing expressible at this level loses mass); *discarding an output marginalizes the kernel*; and *copy is sharing, never resampling* (fan-out of a noise-derived value hands both consumers one realization, while an independent redraw is a second noise port — two different graphs, two different digests, provably different denotations).

For the discrete programs certified in this report, that is the whole sampling story, with no approximation anywhere: a fair coin or a discrete-uniform die is a piecewise-constant map of one noise port with rational thresholds, so the compiled transition rows are exact rationals and the chain *is* the program's law rather than an abstraction of it. Named continuous laws enter by the same discipline, never as new primitives: a *distribution generator* is a packaged symbol whose expansion is a certified piecewise-polynomial quantile table applied to a noise port, carrying an explicit accuracy budget (quantile sup-error ε on a declared core, clipped tail mass declared per side), so a program that draws from a named distribution elaborates — before anything is checked — into the same hypernet language every other certificate speaks. Conditioning is deliberately absent from this kernel vocabulary; §4.6 explains where it lives instead.

4.2 Which distributions: the coverage claim

Distribution coverage is measured against an external census the toolkit does not own: Crooks' *Field Guide to Continuous Probability Distributions* [6], which organizes roughly 170 named forms into about a hundred sections related by an explicit lattice of limits, reparametrizations, and transforms. The MAITRIA book's Markov-kernels chapter closes with a row-per-section coverage dashboard against that census, and its current state supports a strong statement: **probabilistic programs, as hypernets, can draw from any Field Guide family except one.** Every census row is graded certified there — two quantile roots and three wrappers generate most families by composition, with own-table and exact-dispatch routes for the rest, each row carrying its declared budget and a refusal battery — with a single deliberate exclusion: the *unbounded uniform*, which is improper — not normalizable, hence without a quantile function — so the exclusion is an impossibility of the mathematics rather than a limit of table coverage; the book covers the improper form's intended uses with set-valued premises instead.

Two qualifications keep the statement honest. First, a tabled generator is exact about its own inexactness: the generated law couples to the named law within ε except on an event of probability at most the declared clipped mass, so a bound certified for a table-drawing program transfers to the ideal-law program with an explicit, declared correction — never silently. (The discrete campaign of this report is unaffected: none of its cases needs a table, every draw being an exact rational split of its noise port.) Second, the grade: the dashboard's certified rows rest, at this writing, on the toolkit's own test-suite artifacts — accept fixtures at the declared budgets plus their refusal batteries — and the book itself names their migration onto the public reproduction spine as successor work; in this report's vocabulary the coverage claim is **[argued]**, and the dashboard states row by row what each status rests on.

One design note completes the picture at its single exclusion. Where a model wants an uninformative prior over an unbounded numeric range — the role the improper uniform cannot soundly play — a proper, machine-grained substitute is available in design **[aspirational]**: place uniform mass on the IEEE-754 single-precision bit patterns and spread each float's share over the real interval that rounds to it, with overflow mass landing in the two infinity buckets and redistributed inward before renormalization. Because

floating-point spacing doubles at each binade, the induced density falls off reciprocally in the magnitude — the scale-parameter Jeffreys (log-uniform) prior, up to rounding — while remaining a genuine probability measure with the representable range as its support: full support in practice, improper nowhere, and expressible as one more certified table in the dashboard's own format.

4.3 Two judgment shapes, one state-space story

Given the kernel, every number in this report is an instance of one of two judgment shapes, and both are stated with no reference to programs — programs are one producer of chains among many.

The first shape is *finite computation plus a sound remainder*. For a chain presented as stored probability rows with absorbing classes, the checker re-derives the horizon-T distribution exactly and brackets every eventual quantity between what has provably arrived by T and what could still arrive — N_c against $N_c + R$ — with conditioning as a ratio of two such quantities sharing one error budget (§5.1). No termination proof is required, no ergodicity, no spectral gap: the bracket is sound at every horizon beyond T and hence in the limit, and mass the producer failed to expand can only widen it.

The second shape is *contraction, for what finite computation cannot reach*. Moments of absorption times and tail asymptotics sum over an unbounded horizon, so their certificates carry a weight function V and a rate $c < 1$ with a transfer inequality — in its simplest face, $\sum_y K(x,y)\,V(y) \le c\,V(x)$ — recomputed per stored row (§5.2); geometric decay of the V-weighted alive mass then closes the tail in closed form, in $\mathbb{Q}$.

Both shapes are discrete-state in this report's receipts, in two regimes: rows *enumerated* (finitely many alive states, stored outright — the layered and time-homogeneous encodings of §7), or rows *pattern-presented* (infinitely many alive states realizing a declared pattern, stored on a finite window, with the transfer inequality collapsed to finitely many exact inequalities by a template argument — the 3-D walk of §5.3).

The same two shapes are how the mechanism reaches *continuous* state spaces, and the reason the reach is credible is the sampling story of §4.1: because every draw is a piecewise-polynomial function of uniform noise, a continuous-state program is a piecewise-polynomial map of a noise cube, and the objects both shapes need stay polynomial. Reach masses of score-free programs become volumes of polynomial-inequality regions, certified by polynomial sign rows over box partitions with exact box volumes and boundary cells accruing to the upper endpoint — sound by the same one-sided discipline as truncation. Loop tails take the same transfer inequality with V piecewise-polynomial over a box cover of the alive region; for kernels presented by polynomial dilations the transfer integral $\int K(x, \mathrm{d}y)\,V(y)$ is itself a polynomial in x, so the region conditions discharge as exact-rational polynomial-enclosure rows. Density-carrying observations enter on the logarithmic side through certified directed envelopes. This paragraph is design tense throughout [**aspirational**]: the species exist for discrete chains today, the continuous instantiations are designed against the same checker architecture (with development prototypes for the logarithmic-envelope ingredient in the toolkit workshop, one latent deep, outside this report's pinned artifacts), and this report claims nothing about the baselines' continuous or recursive suites (§3).

4.4 The producer strategy

The toolkit's standing production move splits every trusted result into two roles: a *proposer* — fast, approximate, structurally untrustable — and a *disposer* that re-derives, from the candidate data alone, everything the certificate asserts, in exact rational arithmetic, and accepts or refuses. Nothing the proposer computed appears in a trusted conclusion. What follows locates each ingredient of this campaign on that boundary.

Exact computation (no search). The horizon-T masses are never approximated: the producer runs an integer-scaled dynamic program over big integers (one division at readout) on the layered or time-homogeneous chain, so class masses, rejected mass, and residual are exact rationals by construction. The checker repeats the same evaluation from the stored rows; the producer's only way to be wrong about a mass is to claim a bracket the recomputation refutes.

Search, tier one: encoding and horizon. The producer chooses the chain presentation — layered, one layer per step; or time-homogeneous with states materialized once, which buys orders of magnitude of horizon under the same checking budget; or pattern-plus-window where the alive set is infinite — along with any exact lumping the program's symmetries admit (the coupon collector's exchangeability) and the unrolling depth. These choices are the producer's freedom in the same sense that interior-point settings are Diabolo's: disclosed in the receipts, priced into the wall-clock, and irrelevant to soundness — a poor encoding yields a wide bracket or a refusal, never a false verdict. Lumpings, like every translation step, are outside the trust path and covered by the differential batteries of §6.3.

Search, tier two: contraction templates. Weight functions V, rates c, and region data are proposed by whatever works, and this campaign's receipts deliberately span the spectrum: a closed-form left eigenvector (the coupon collector, whose rate 4/5 is the alive spectral radius); eigen-tight regional weights with a unit floor (the Israeli–Jalfon ring, where sitting on the spectral radius makes the certified tail bound achieved rather than merely sound); and, where a whole template family is provably empty (the 3-D walk's pure geometric family, §5.3), a new family designed at its general form and searched over finitely many shape parameters. Numeric proposals — a float linear solve, a damped power iteration — are equally admissible: the checker recomputes the transfer inequality per stored row and cannot tell, and does not care, where a template came from.

Search, tier three: programs that search. The outer loop is itself programmable: bound-search scripts — written by a language model against the checker's verdict vocabulary — choose encodings, depths, template families, and refinement steps, reading refusals and wide brackets as data. This is the architecture's point rather than an implementation convenience: because the only trusted component is the checker, arbitrarily clever search — floating-point, learned, generative — costs zero trust, and a wrong search program costs a rejected certificate, never a false verdict. In this campaign the scripts were driven case by case; packaging the loop behind a persistent agent-session surface is design-tense work **[aspirational]**.

4.5 Loops: the exact support boundary

Both baselines exist because loops are where guaranteed bounds get hard, so this report states its loop support exactly — first what the shipped species handle, then the boundaries, each with its reason.

Supported.

1. **Unbounded `while` loops with no termination assumption**, for mass and posterior queries: the residual construction needs neither almost-sure termination nor any bound on running time — nonterminating or unexpanded mass accrues to R and widens the bracket, never falsifies it.

2. **Loops whose return value records the running time**: the layered encoding lets stored rows depend on the step index, so an absorption-time query variable rides the class labeling without a product state space (the geometric counter, the asymmetric walk, the die paradox).

3. **Nested, sequenced, and step-dependent loop structure, in kind**: the certificate species consume stored-row chains and never see loop syntax — a nested or sequenced loop is a chain over the joint configuration space, and a step-dependent body is native to the layered form. Every case exercised to date is a single loop or a loop-free net, so nesting is supported in kind rather than by receipt.

4. **Unbounded behavior, two ways**: unbounded *time* over finitely many alive configurations (the time-homogeneous encodings — the token ring, the coupon collector, the Israeli–Jalfon ring, the Herman ring — where a horizon of 10^3 costs little because states are materialized once); and unbounded *space*, pattern-presented with a stored finite window and a template-collapsed contraction (the 3-D walk, whose alive-reachable set is infinite; this is the development-oracle lane of §9.1).

5. **Deterministic prologues**: initialization before the loop enters moment claims through the anchor-offset datum, with the prologue's determinism recomputed from stored rows (§5.2), never subtracted out of band.

6. **Hard conditioning inside loops**, per §4.6: a rejecting observation in the loop body is one more row destination (the die paradox).

Not supported, each boundary with its reason.

1. **Moment and tail queries demand more than mass queries do.** Truncation is sound for brackets on masses and posteriors, but a first moment sums over the whole horizon: the stored rows must be a total law on the alive-reachable set (a reachable state without a stored row is a refusal for moment claims, not a widening), and a verified contraction with $c < 1$ must exist. The current tail grammar is geometric only — a pattern-presented chain whose absorption-time tail is genuinely subgeometric would be refused rather than certified with a weaker tail form.

2. **Rows have finite support.** A single step may not draw from an unbounded-support distribution directly; such draws are written as loops of bounded draws — their usual program form — which moves the unboundedness into the time dimension, where the residual and contraction machinery own it.

3. **Continuous state in the loop is not claimed.** The mechanism is designed (§4.3); the baselines' continuous-recursive suites are named future targets; no continuous loop appears in this report's receipts.

4. **General recursion and higher-order functions are not modeled.** The species consume chains, and a chain has no call stack. A first-order loop program is a chain over its variable valuations; a recursive program with countable reachable configurations could in principle be presented the same way (stack contents as state), but no producer for that presentation exists and no such case is exercised. GuBPI's universal-language suite is correspondingly out of scope here.

5. **The query vocabulary is eventual absorption**: return-value histograms, posteriors over them, and absorption-time moments and tails, with class states absorbing (recomputed, never assumed). Fixed-horizon marginals of transient states, though the same evaluation computes them, are not a claim row in the species today.

4.6 `score`: the exact support boundary

Neither `observe` nor `score` is part of the Markov-kernel language, and within that fragment the omission is a theorem: in the dilation reading of §4.1 every kernel is normalized — total boxes, probability reference measure — so there is nothing at that level for a weight to multiply. The fragment is not the last storey, however. The designed tower continues the way it began: just as the Markov-kernel fragment extends the polynomial fragment, a *probabilistic-program fragment* extends the Markov-kernel fragment with `score` and `observe` as generators — the tier where unnormalized weights are native rather than excluded [aspirational; its design rides the continuous-lane work in §9.1]. In the shipped system, conditioning enters the *certificates*, in three graded ways, each with its exact boundary.

Hard conditioning: shipped, as a class rather than a score. A failed `observe` absorbs into a distinguished reject class; every stored row remains a plain probability row, and the posterior is a correlated ratio of absorption measures with the rejected mass excluded from the normalizer (§5.1): bracket $[\, N_c/(S{+}R),\ (N_c{+}R)/(S{+}R)\,]$, refusal when no accepting mass has been witnessed ($S = 0$), width class-uniform at $R/(S{+}R)$. The die paradox and the burglar alarm are this mechanism's receipts, and observation-free programs degenerate gracefully: evidence plus residual is one, and posterior brackets coincide with unnormalized ones (the token ring).

Bounded scores in $[0,1]$: expressible today, by rejection. A soft score with rational weight $w \in [0,1]$ is one probability row away: branch to the reject class with probability $1-w$ and continue with probability w. The accepted-trace measure is the score-weighted measure — the rejection-sampling reading of the same program — so the posterior is unchanged, and at these weights hard and soft conditioning differ only in program text. Two limits, stated so the boundary is sharp: weights above 1 do not rescale into this form in general (dividing by a common bound preserves the posterior only when every trace applies the same number of scores), and weights that are values of a continuous density at a continuous latent are outside it entirely.

General scores: the score face of the contraction species, design tense. The strengthened contraction rule of §5.2 carries a per-state score $s(x)$ inside its transfer inequality ($\sum_{y\in A} s(x)K(x,y)V(y) \le c\,V(x)$ on the region): scores ride the *claim rows* as certificate data while every stored row stays a probability row, and the checker re-derives score-weighted masses from plain rows times declared scores — the kernel language never learns about weights. The Lean status of that rule's lemma family, score face included, is as stated in §5.2; the assembly that would turn it into posterior brackets for score-carrying programs at unbounded horizon — the score-weighted analogue of the correlated ratio — is designed but is not part of any row in this report (development prototypes exist in the toolkit workshop; no pinned artifact) **[aspirational]**. Density-valued scores on continuous latents are the standing residue beyond that, part of the continuous lane of §4.3. For contrast: GuBPI treats `score` natively, as an interval weight accumulator over interval traces — exactly the generality the shipped species here trade away for exact arithmetic and a checkable artifact, and the trade is stated so the reader can weigh both sides.

5 What the benchmarks forced: the certificate species

The campaign's durable output is not the comparison table; it is four additions to the specification calculus, each forced by a case that no existing rule covered, each designed at its general form rather than the benchmark's minimal shape. This section describes them in the order the cases forced them. Soundness status, uniform across the section: the absorption-bracket and contraction lemmas named below are machine-checked in Lean 4 — sorry-free, axiom closure exactly [`propext`, `Classical.choice`, `Quot.sound`], re-audited on every build by probe modules — and are built on the toolkit's existing mechanized chain semantics via an explicit duality bridge (forward mass evolution equals the endpoint marginal of the mechanized path law), so the new brackets ground in the same semantic universe as the rest of the calculus rather than in a parallel formalization.

5.1 Finite-horizon absorption brackets (`DtmcAbsorb`)

The general judgment behind residual-mass semantics, stated with no reference to programs: given a chain presented as stored probability rows, a class labeling on absorbing states, and a horizon T, the checker recomputes the T-step distribution $m_{t+1} = m_t K$ from the stored rows and certifies, for each class c, the bracket $[N_c,\ N_c + R]$ on the *eventual* absorption measure of c — where N_c is the class-c mass at T and R the still-alive (residual) mass. Monotone absorption plus conservation make the bracket sound for every $t' \geq T$ and hence for the limit. Programs are one producer of such chains among many; nothing in the species mentions them.

Two design points carry most of the value:

1. **Truncation is sound by construction.** The evaluation semantics treats any state without stored rows as absorbing (unclassified). Mass reaching an unexpanded state therefore accrues to the residual R — it can widen a bracket, never falsify one. A whole side-condition class dissolves: a standalone checker for this mathematics must patrol closure ("does the stored row set cover everything reachable?"), and an earlier in-house prototype did exactly that, mutant battery and all. In the geolog species the corresponding attack is *unrepresentable* rather than checked for. Moving the mathematics into the stronger calculus made checking structurally cheaper — the opposite of the usual expectation, and worth recording as a data point about certificate design.

2. **Hard conditioning is a class, not a score.** Rejected runs (a failed `observe`) absorb into a distinguished reject class. Every stored row remains a plain probability row — no weighted-chain machinery on the histogram path — and the posterior becomes a *correlated ratio* of absorption measures: with S the total classed-accepting mass at T, the posterior bracket for class c is $[\,N_c/(S{+}R),\ (N_c{+}R)/(S{+}R)\,]$, refusing when $S = 0$ (no positive evidence witnessed). Because numerator and denominator move together, this is never worse and often strictly sharper than dividing independent intervals — the die-paradox row below is the first live receipt, at 3.0× — and the bracket width is *class-uniform*, exactly $R/(S{+}R)$: the tightness of the entire certified histogram is one number.

A third observation fell out of the comparison work: for (sub)stochastic chains, the conservation residual (alive mass at T, computed exactly by the same dynamic program) is tighter than any contraction-invariant bound on the same quantity and needs no invariant at all. Contraction certificates are therefore *not* histogram machinery; they are the cargo of expected values, higher moments, tail asymptotics, and (in future work) soft scores — which is the next species.

5.2 The contraction strengthening (`QGenFormContraction`)

The expected-value and tail cases (§8) need geometric control of the alive mass. The calculus already had an incumbent rule for nonpositive-drift generator forms on chain content; the campaign's choice — deliberate, and worth stating as method — was to *strengthen the incumbent* rather than add a sibling: a score-carrying, region-restricted contraction form whose certificate data are a weight function V (stored on a region A), a contraction constant $c \in [0, 1]$, and an exit-domination constant κ, with the checker recomputing per stored row that $\sum_{y \in A} s(x)K(x,y)V(y) \leq c\,V(x)$ on A, that no re-entry into A occurs, and that exits are dominated by $\kappa V(x)$. Conclusions: weighted alive-mass decay $\langle m_t|_A, V\rangle \leq c^t \langle m_0|_A, V\rangle$; and for $c < 1$, closed-form geometric and first-moment tail bounds in $\mathbb{Q}$. At $s \equiv 1$, $c = 1$, $A =$ all states, the strengthened rule's consumption re-derives the incumbent rule's exact statement — a *subsumption* proven in Lean as a mechanized implication chain, so every certificate admitted under the incumbent remains valid under the strengthening. One rule now stands where two could have accumulated.

Two general facilities landed inside this species because specific cases demanded them:

- **The anchor-offset datum** (forced by Israeli–Jalfon): programs whose initial samples ride a deterministic prologue of length k have absorption time $T = \mathit{count} + k$. The offset is certificate *data*, and the checker recomputes the prologue's determinism from the stored rows ($P[T > k{-}1] = 1$; refuse otherwise) and starts the partial moment sum at $t = k$ — never an out-of-band subtraction. Both directions carry mutant teeth: an early offset is refused at the determinism recomputation; a dropped offset fails the bracket.
- **Pattern-presented chains with ramp-profile templates** (forced by the 3-D walk, §5.3).

Status, stated precisely because the grading depends on it: the Lean lemma family for this species (weighted core with the nonnegativity hypothesis on V *dropped* — strictly stronger than its own specification — score face, regional packaging, exit domination, series identities, both tail assemblies, and the $c{=}1$ subsumption bridge) is proven and landed. The *schema realization* — the rows as sealed theory content — landed at the revision event for the finite face: expected-value verdicts for the finite-region cases now ride container content checked against the sealed contraction closure, flag-free. The 3-D case's pattern face was deliberately deferred at the event's adversarial review (§9.1), and its verdicts still ride the development oracle (an exact-rational checker for the draft face, run beside the container lane, with the certificate bound to the same stored rows by hash). The tables mark that row accordingly.

5.3 Why ramp profiles: a template family with an emptiness proof

The 3-D asymmetric walk (the baseline suite's slowest solved case) is time-homogeneous with an *infinite* alive-reachable set, so per-row enumeration cannot cover it; the certificate presents the chain as a pattern (dimension and up-probability as species data) with stored rows on a finite window, and the contraction certificate must collapse infinitely many row conditions to finitely many exact inequalities. The natural first template — per-coordinate geometric weights $V(x) = \prod_i \gamma^{x_i}$ — provably cannot work: with saturating decrement, a zero coordinate bounces upward when selected, and the two resulting constraints reduce to $(\gamma - 1)^2 < 0$. The pure geometric family is *empty* for this walk — which is precisely why the baseline's own optimizer needed degree-3 polynomial invariants and 151 seconds on the same case. The family that works: per-coordinate *ramp profiles*, flat (ratio exactly 1) near the origin and rising to a geometric rate in the far field, with a class-collapse side condition $((d-k)\,r_{\max} + k\,r_\infty \le d\,c$ for $k = 1..d$) that is generic over the whole d-dimensional uniform-axis saturating-walk family and subsumes the one-dimensional template used by the coupon-collector case as $d = 1$. The emptiness argument and the collapse condition are the case's contribution to the calculus; the benchmark row is just their receipt.

5.4 A revision-event passenger, identified and landed

The Beauquier token-ring near-miss (§8.5) had a known repair that rode the absorption species' revision event as a strengthening of the same rule, not a sibling: a per-class *reachability sharpening* — residual mass is charged to a class only from states that can still reach it through strictly positive stored rows — which closes that case's bracket to width exactly 0 at $T \ge 1$ under strictly weaker hypotheses than the baseline's printout leans on (no termination assumption). The first edition wrote this paragraph in the future tense because it did not exist yet. It exists now, machine-checked beyond its own promise: the sharpened upper bound needs *no hypotheses on the class at all* (the well-formedness demands protect only the monotone lower side); the old uniform-residual bound is its one-line corollary; a kernel-checked strict-improvement witness (unreachable class: incumbent width 1, sharpened width 0) re-checks on every build; and old certificates' looser claims remain admissible, so nothing already emitted was invalidated. The tightness table carries its receipt — twice over: the token ring's row, and (in the wave that landed beside this event) a second protocol case — the Herman ring of §7 — whose queried class is likewise unreachable from the alive mass, closed to a width-zero point by the same rule revision with zero additional theory motion. One rule revision, two cases converted: the benchmarks-as-forcing-functions loop, closed.

6 Protocol: checking, translation, measurement, preregistration

6.1 The checking path

Published verdicts come from the geolog fast-path engine with the QTSL plugin: the producer seals each certificate as a content-addressed container (in this campaign, 0.7–20 KB each; 5 to 201 claims per container), and the checker validates it against the specification theory's closure — since the revision event, the *sealed* closures themselves (the absorption species' closure at 12 theories, 747 axioms; the contraction species' at 13

theories, 753 axioms; content-addressed, double-build byte-equal identity, resolved from the checker's own pinned roster with no flags) — with expected-claim-count gates, refusing on any malformed, unbindable, or arithmetically unsupported claim. Every case ships a designed-locus mutant battery through the same binary: brackets tightened past the truth must FAIL, widened brackets must ACCEPT (positive controls), byte-flipped containers must REFUSE, well-formedness violations must REFUSE at their named locus. Across the nine cases the batteries comprise roughly seventy-five negative loci plus positive controls, all verified in class by two recheck drivers that re-run every committed container and mutant end-to-end.

6.2 Anatomy of one certificate

Auditability is the product, so here is what a container actually holds, on the smallest case (the die paradox; 13 claims; sealed-lane check ∼50 ms, resolved from the checker's pinned roster with no flags). The certificate content is a query entity — chain reference, horizon, class labeling including the distinguished reject label, anchor state — plus the stored rows themselves: the initial distribution, one exact-rational transition row per stored state (row masses checked against the (sub)stochastic condition), and the class/reject labelings. The claim rows are: one well-formedness claim (every classed or rejected state is stored-absorbing — recomputed, never taken on faith), one absorption-bracket claim per class, and one posterior-bracket claim per class. That is all. The masses N_c, S, and R of §5.1 appear nowhere in the container: the checker re-derives them from the rows by the T-step evaluation, and the posterior comparisons are performed division-free (cross-multiplied), so no rational division sits on the verdict path. Evaluation budgets (state count, horizon × row count) refuse oversized queries loudly rather than degrading.

Three verdict classes, kept distinct because they mean different things: **ACCEPT** (every claim discharges), **FAIL** (a claim's own arithmetic refutes it — the verdict a tightened-bracket mutant must draw), and **REFUSE** (malformed, unbindable, over-budget, or zero-evidence content — the checker declines to adjudicate rather than guessing). Positive controls — widened brackets, rescaled weights — must ACCEPT, and do; a battery whose mutants all refuse proves nothing about a checker that refuses everything, which is why the controls are part of every case's battery.

6.3 Translation validity

"We match benchmark b" presupposes our chain *is* b's program. Per case: (a) the source program is vendored verbatim (MIT; the original comparison trio additionally diffed byte-identical against the Zenodo artifact); (b) the chain encoding is documented beside it; (c) a Monte-Carlo differential simulates the source program's published semantics directly (rejection sampling for hard observations) for 10^6 paths and compares against the chain's exact masses at 4-sigma binomial bands — with *cross-program* teeth: deliberately mis-encoded variant chains (a wrong guard, a synchronous-update slip) are refused by the same differential at $>4\sigma$; (d) exact ground truth is contained in every emitted bracket wherever a closed form or an exact oracle exists (seven of the nine cases; the remaining two have exact linear-solve or deep-truncation oracles). The honest boundary, stated: nothing binds the container's chain rows to the source program's text *inside the wire* — the binding is exactly these receipts, and the translation is itself untrusted producer output whose mutant battery is (c) and (d).

6.4 Measurement protocol

All measured runs, both sides: one machine — an NVIDIA GB10 (Grace, aarch64, 20 cores) — pinned to a single Cortex-X925 core at 3.9 GHz (heterogeneous cores make unpinned single-thread timing bimodal), absolute niceness 19 with the scheduler value read back from `/proc` and recorded per block (a claimed niceness without read-back is a command-line fact, not a measurement), one discarded warmup then medians of ≥ 5 runs (20-run harness for millisecond-class cells), load averages recorded before each block, every raw stdout and resource record committed. House walls are end-to-end *including certificate checking* (producer → container emit → fast-path check, plus the development-oracle check on expected-value cases); baselines are charged their own process end-to-end, with their internal self-timings kept as context (Diabolo's published millisecond numbers are internal; its process adds 1–2 ms of startup; GuBPI's internal excludes about 0.18 s of .NET startup — both directions reported).

Baseline configurations are the artifacts' own. Two findings from the configuration archaeology, both material: (1) the Diabolo repository's `comparison.sh` carries *different, tighter* settings than the published GuBPI-comparison table used; the table's actual commands (unrolling 100/14/6 at default limit — "configured to produce the same bounds" means Diabolo dialed *down* to GuBPI's output quality) were recovered from the Zenodo artifact README after a per-case time ordering contradicted the published row. Both

configuration families were measured and banked; comparisons below use the published-table commands. (2) GuBPI does not successfully normalize the die-paradox case (its normalized bounds are vacuous) — documented in the baseline's own artifact README, not an artifact of our rebuild — so that row compares unnormalized lower bounds, per the artifact's own practice.

Build provenance, disclosed: Diabolo built clean (two system solver libraries installed); GuBPI required a framework bump and a dependency bump to obtain linux-arm64 native solver binaries (root-caused to its LP wrapper's pinned solver vintage shipping no arm64 natives; patches preserved as diffs beside the receipts; solver backend unchanged). Consequence stated honestly: our GuBPI runs a 2026 toolchain on aarch64 versus the paper's older toolchain on unstated x86 hardware. The same-box comparison is unaffected — both tools carry current toolchains on the same machine — and published numbers were context only. For calibration: our box reproduces Diabolo's published speedup pattern over GuBPI on the shared trio and *widens* it (449×/69,500×/266,000× internal-vs-internal, against published $1.2\times10^2/4.5\times10^4/2\times10^5$) — the baselines run 2–6× faster here than on their papers' hardware, which is exactly why same-box measurement is doing real work.

6.5 Preregistration

Every case carries a preregistration committed to the repository before its runs: the claim, the checker and expected verdict loci, the mutant list with each mutant's predicted failure class, the tightness target, and the wall-clock target. Deviations are reported as deviations in the case receipts, never absorbed. Three are worth the reader's attention, because each is the discipline working rather than failing:

1. **A preregistered derivation was falsified by the campaign's own oracles.** The Israeli–Jalfon preregistration derived the expected stabilization time by hand as 41/28, lumping states by token count under ring symmetry — and preregistered a *tension*: the hand value sat outside Diabolo's published bracket, so one of the two had to be wrong, and both resolutions were named in advance. The mechanical oracle chain (exact linear-solve, 10^6-path differential against the vendored source, one-step direct transcription over all alive configurations) falsified the hand derivation: the source program's sequential update blocks break ring symmetry (two configurations with the same token count have different expected times), and the truth is 373/256 — inside Diabolo's bracket. **The preregistered tension resolved against our own derivation and confirmed the baseline's soundness on this case.** The final certificate's upper endpoint is exactly 373/256 (§8.1).

2. **The configuration pin was corrected by evidence, twice, in the open.** The first-blood preregistration pinned its claims to "the published comparison configuration"; identifying what that configuration actually was took two rounds of artifact archaeology (§6.4), each recorded as a correction to the preregistration's concrete parameters with the abstract pin unchanged.

3. **Wall-clock targets were missed and are reported as missed.** The preregistrations set $\geq 10^2\times$ against GuBPI per case and within-10× of Diabolo's residual lane at first blood; §8.5 scores the misses.

The preregistrations also made one structural prediction — that at least one row family would demand a genuinely new certificate rule rather than an encoding of an existing one — which held (§5): the absorption species is new, and the contraction species is a proven strengthening of an incumbent.

7 The nine cases: programs, encodings, ground truth

Each case below names its vendored source, the chain encoding the producer emits, and the ground truth the certificate must contain. The encodings are documented at greater length, with their oracles, in the reproduction spine's `PROGRAMS.md`; this section is the reader's map. A recurring structural note: cases 1–3 are *layered* encodings (finite truncation, one layer per step); cases 5–9 are *time-homogeneous* (states materialized once, stored rows independent of horizon — the token ring is 35 states and 91 rows whether the horizon is 51 or 1000); case 8 is time-homogeneous over an *infinite* alive set, presented as a pattern with a stored finite window; and case 9 is the one program both baselines' suites contain, certified for both tools' queries from a single container.

1. Geometric counter. `Y := 0; while flip(1/2) { Y += 1 }; return Y`. One alive state per layer; absorb into class Y with probability 1/2 per step. Truth: $P[Y{=}n] = 2^{-(n+1)}$, $\mathbb{E}Y = 1$.

2. Asymmetric random walk. `X := 1; C := 0; while X != 0 { C += 1; if flip(1/4) { X += 1 } else { X -= 1 } }; return C` (adapted by the Diabolo authors from Klinkenberg and colleagues [4]). Alive states are positions $x \geq 1$; the query variable equals the absorption time; absorption occurs only at odd times. Truth: $\mathbb{E}C = 2$; per-class masses are ballot-walk rationals the dynamic program recovers exactly; tail $\Theta(n^{-3/2}(\sqrt{3}/2)^n)$.

3. Mossel's die paradox. Throw a fair die until a 6, `observe` after each throw that the face is even; count throws. An odd face rejects the run — hard conditioning; the reject class carries that mass and the posterior normalizer excludes it (§5.1). One alive state; per transition: absorb-accept w.p. 1/6, continue w.p. 2/6, reject w.p. 3/6. Truth: $P[T{=}n \mid \text{acc}] = (2/3)(1/3)^{n-1}$, $\mathbb{E} = 3/2$, evidence $Z = 1/4$.

4. PSI burglar alarm. The classical finite Bayes net (earthquake, burglary, alarm, phone, waking) with a hard observation, from the PSI suite via GuBPI's benchmarks. Layered encoding, one sampled variable per layer; the final layer folds the observation test, absorbing satisfying assignments into their `burglary` class and rejecting the rest. The program is finite, so horizon 4 is *full unrolling*: the alive frontier is empty, the residual is exactly zero, and the class-uniform width $R/(S{+}R)$ degenerates to 0 — both posterior brackets are exact points. Truth: PSI's exact $P[\text{burglary} \mid \text{called}] = 2969983/992160802 \approx 0.0029934$, which the producer's dynamic program reproduces bit-exactly and an independent 16-assignment enumeration oracle re-derives without the chain machinery.

5. Beauquier–Gradinariu–Johnen token ring, 3 processes. From GuBPI's protocol-estimation suite: three processes each carrying two fair bits, one synchronous round per loop iteration reading old values throughout; the loop runs while the p-bit sum is ≥ 2; the query is $P[\mathit{count} < 1]$ — the initial sample already landing at p-sum ≤ 1. Time-homogeneous encoding: a start state whose row is the 64-outcome initial sample, 32 alive ring configurations, absorption into two count classes; score-free, so evidence plus residual is exactly 1. Truth: exactly 1/2 (binomial count on three fair bits). The bracket is $[1/2,\ 1/2 + R_T]$ with R_T the exact alive mass at the horizon — the lower endpoint *is* the truth; only the sound residual sits above it (§8.5). Translation extras here: a one-step direct-transcription oracle over all 32 alive configurations, and a cross-program tooth — a sequential-update slip variant (guards reading already-updated neighbor bits) emits an internally-green certificate and is *refused* by the Monte-Carlo differential at 4σ, which is exactly why the differential is part of the battery.

6. Coupon collector, five coupons. Draw uniformly among five coupons until all have been seen; the query is the expected number of draws. Time-homogeneous: exchangeability lumps the five flags to the count $k \in 0..4$ of distinct coupons (a translation step, discharged by the five-explicit-flags differential and an inclusion–exclusion oracle); from k, a new coupon w.p. $(5{-}k)/5$. The expected-value certificate is a survival-form assembly: exact partial sums from the dynamic program plus a geometric tail from template rows at $\gamma = 4/5$ — the alive spectral radius itself, so every transfer inequality holds with equality and the certified tail is asymptotically the true tail. Truth: $\mathbb{E} = 5H_5 = 137/12$.

7. Israeli–Jalfon self-stabilization, ring of four. Four token flags sampled iid fair; while more than one token remains, an iteration snapshots the flags and walks the four if-blocks *in order* — a set old flag clears itself and sets a uniformly chosen neighbor. Time-homogeneous over the 16 bit-vectors plus a start state carrying the Bernoulli$(1/2)^4$ prologue as one row; token count ≤ 1 classifies into the stabilized terminal class. The chain's absorption time is $\mathit{count} + 1$ on every path, so the expected-value certificate carries anchor-offset 1 (§5.2) with the prologue's determinism recomputed from stored rows. Truth: $\mathbb{E}[\mathit{count}] = 373/256$ by exact linear solve over the 11 alive configurations — the sequential block order breaks the ring's rotational symmetry, which is what falsified the preregistered class-lumped derivation (§6.5). Alive spectral radius exactly 1/2.

8. 3-D asymmetric random walk. From $(1, 1, 1)$: choose an axis uniformly, increment it w.p. 1/4, else decrement with saturation at zero (the source language's naturals); count every step; absorb at the origin. The query is the expected absorption time — the baseline suite's slowest solved row. Time-homogeneous with an infinite alive-reachable set: the certificate presents the chain as a pattern (dimension, up-probability) with stored rows on a finite window; the mass bracket rides the absorption dynamic program, and the expected-value tail rides ramp-profile template rows (§5.3) collapsed to finitely many exact-rational inequalities. No closed form exists; the float linear-solve oracle gives ≈ 16.69099, the 10^6-path differential agrees, the exact horizon-200 bracket is $[16.690028724,\ 16.702938972]$, and the brackets nest across horizons.

9. Herman's self-stabilization ring, 3 processes. The comparison set's one program that *both* baselines run: Diabolo's suite carries it (via the PSI benchmark collection) and queries the expected self-stabilization time; GuBPI's probability-estimation suite carries the same protocol and queries $P[\mathit{count} < 1]$. Three bits sampled iid fair; while the bits sum to anything but one, a synchronous round — each process compares its own old value with its ring predecessor's old snapshot, resampling fair on equality and copying the predecessor otherwise — increments the counter. Time-homogeneous encoding in the token ring's style: a start state carrying the Bernoulli$(1/2)^3$ prologue as one row, the eight bit configurations, and absorption into two count classes, with class $\mathit{count} < 1$ fed by the start transition alone; score-free, so posterior brackets coincide with unnormalized ones, and the chain's absorption time is $\mathit{count} + 1$ on every path, so the expected-value certificate carries anchor-offset 1 (§5.2). One container serves both baselines' queries, and one pipeline wall prices both rows. Truth: $\mathbb{E}[\mathit{count}] = 4/3$ by exact linear solve over the five alive configurations (matching a comment in the vendored source itself); $P[\mathit{count} < 1] = 3/8$, binomial; and the count tail exactly, $P[\mathit{count} > n] = (3/4)(1/2)^n - (1/8)(1/4)^n$, with alive spectral radius exactly $1/2$. Where the Israeli–Jalfon source updates sequentially, this round is genuinely synchronous in the source (every later block reads old snapshots), so the cross-program differential tooth is the inverse slip: a variant chain with the snapshots dropped emits an internally-green certificate and is refused by the Monte-Carlo differential at $>4\sigma$.

8 Results

Nine cases. The first three are the shared trio on which the Diabolo paper compares itself against GuBPI — the natural first table, run at the published comparison configuration. The remaining six extend the comparison to hard conditioning (burglarAlarm), a distributed-protocol reachability case (Beauquier–Gradinariu–Johnen), three expected-value/tail cases from Diabolo's own suite including its slowest solved row, and one program — Herman's self-stabilization ring — present in *both* baselines' suites, certified from one container against each tool's own query. "n/a" means the case is absent from that tool's suite — verified against the vendored benchmark tree and stated, never interpolated (the Herman ring, present in both suites but queried differently by each, gets a pointer between its two rows instead). All house absorption rows and the coupon-collector, Israeli–Jalfon, and Herman-ring expected-value rows **[shipped]**, checked flag-free against the sealed revision (§9.1); the 3-D expected-value row carries the development-oracle lane marker (*ev-oracle*) — **[argued]**, pending its pattern face's own revision event.

8.1 Bound tightness

case (query)	this work (exact $\mathbb{Q}$; width)	Diabolo same-box	GuBPI same-box
geometric counter (posterior bins, u=100)	7.889×10^{-31}	7.89×10^{-31} — **tie** at print precision	orders wider (its Z interval width $\sim 10^{-15}$; per-bin bounds far above both)
asymmetric random walk (posterior, u=14)	7.342×10^{-3} (= 123181/16777216 over evidence+residual)	7.40×10^{-3} — house **strictly tighter** (exact rationals vs. outward-rounded float)	Z-width 0.639
die paradox (posterior, u=6)	5.464×10^{-3} (= 1/183)	1.65×10^{-2} — house **3.0× tighter** (correlated ratio vs. interval quotient)	normalized bounds vacuous upstream (artifact-documented); unnormalized lower bounds compared per its own practice
burglar alarm (P[burglary | called], full unrolling)	**exactly 0** — point posterior 2969983/992160802	n/a	0 at print precision (validated float; its Z interval width 6.8×10^{-16}) — **win-or-tie**
Beauquier ring, 3 proc. (P[*count*<1]; u=51 / u=1000)	**exactly 0** at both horizons — the point $[1/2, 1/2]$, exact $\mathbb{Q}$, via the reachability-sharpened residual (§5.4); no termination assumption	n/a	prints the exact point $[0.5, 0.5]$ via score-free Z=1 (a float printout leaning on Z=1 by construction) — **tie at the point, on strictly weaker hypotheses** (first edition: a near-miss by 2.5×10^{-101}; §8.5)
coupon collector, 5 (EV, u=80)	4.417×10^{-7}; 137/12 contained	1.426×10^{-1} (= its published row) — house $3.2 \times 10^{5}\times$ **tighter**	n/a
Israeli–Jalfon, ring of 4 (EV, u=31)	1.441×10^{-9}; upper endpoint **exactly the truth** 373/256	3.983×10^{-4}, contains 373/256 (soundness confirmed) — house $2.8 \times 10^{5}\times$ **tighter**	n/a
3-D asymmetric walk (EV; horizons 200/150)	1.291×10^{-2} / 1.117×10^{-1} *(ev-oracle)*	1.19×10^{6} same-box, byte-stable at the file's own flags (published face 6.85×10^{5}; optimizer-sensitive) — house $9.2 \times 10^{7}\times$ **tighter**	n/a
Herman ring, 3 proc. (EV, u=31)	1.397×10^{-9}; upper endpoint exceeds the truth 4/3 by exactly $(1/6)\,4^{-30}$ (mechanism below)	4.750×10^{-4}, contains 4/3 (its published row renders $[1.333, 1.334]$) — house $3.4 \times 10^{5}\times$ **tighter**	— (runs this program, but queries it as P[*count*<1]; next row)
Herman ring, 3 proc. (P[*count*<1]; u=31 / u=101)	**exactly 0** at both horizons — the class-zero absorption bracket is the point $[3/8, 3/8]$, exact $\mathbb{Q}$, via the reachability-sharpened residual (§5.4); no termination assumption	— (queries this program as the expected value; row above)	prints the exact point $[0.375, 0.375]$ via score-free Z=1 — **tie at the point, on strictly weaker hypotheses**: the sharpening's second conversion, landed before this row ever stood as a loss

Truth containment holds everywhere a ground truth exists: closed forms on the shared trio, PSI's exact posterior on burglarAlarm, 1/2 on the token ring, 137/12 and 373/256 on the coupon collector and the Israeli–Jalfon ring, 4/3 and 3/8 on the Herman ring (where both baselines' own brackets were confirmed sound as well), and nested-bracket/oracle agreement on the 3-D walk. On every case either baseline shares with this work, the verdict on the tightness axis is **win or tie**, and no tightness loss stands: the first edition's one loss, the token-ring near-miss, was converted by the rule revision (§8.5), and the Herman ring's protocol row is that revision's second width-zero receipt.

Three rows deserve their mechanism named. The die-paradox win is structural, not arithmetic: at identical truncation, the correlated ratio (numerator and denominator sharing the residual) beats the interval quotient by construction; 3.0× is that theorem's first same-box receipt. The Israeli–Jalfon row's upper endpoint is not merely tight but *exact*: the certificate's per-state weights make every transfer row an equality — the contraction rate sits on the alive spectral radius 1/2, and the heavier states are transient within three iterations — so the geometric tail bound is achieved and the bracket's upper end *is* 373/256. An exact-rational certificate can sit on the spectral radius; a float LP reports 0.5011. The Herman-ring expected-value row is the same certificate family in the opposite regime: its weights also make every transfer row an equality with the rate exactly on the spectral radius (1/2), but its above-floor states *recur* rather than dying out, so the geometric tail bound is sound without being achieved and the upper endpoint exceeds 4/3 by exactly $(1/6)\,4^{-30}$ — derived by hand in the preregistration and confirmed by the oracle chain to the last bit. The pair witnesses both sides of eigen-tight-versus-achieved.

8.2 Wall-clock, end to end

House walls include certificate checking; baselines are their own process end-to-end at their own configurations; internal self-timings in parentheses as context. House cells were re-measured after the revision event through the flag-free checking lane (same protocol, same box; the receipts record a higher ambient load than the first edition's runs, honestly): the closure re-validation retired where it dominated (the token ring, −44% to −50%), while the many-class cases pay a few added milliseconds for the sharpened per-class reachability recompute — both directions reported, the mechanism split isolated by a same-load control in the receipts.

case	this work	GuBPI	Diabolo	verdict
geometric counter	0.298 s	0.880 s (0.701)	2.67 ms (1.56 ms)	3.0× under GuBPI — preregistered 10^2 **missed**; **above** Diabolo (§8.5)
asymmetric walk	0.108 s	51.62 s (51.40)	1.76 ms (0.74 ms)	**480**× under GuBPI; **above** Diabolo
die paradox	0.105 s	74.76 s (74.56)	1.18 ms (0.28 ms)	**713**× under GuBPI; **above** Diabolo
burglar alarm	0.118 s	0.260 s (0.080)	n/a	**2.2**× **win** (preregistered 2–10×); GuBPI's internal 0.080 s beats our end-to-end — internal times are context, and this row is why that rule cuts both ways
Beauquier ring	0.132 s (u=51) / 0.803 s (u=1000)	7.87 s (7.67)	n/a	**60**× / **9.8**× — the sealing's largest wall delta (−44% / −50%: the per-process closure re-validation retired exactly where it dominated); preregistered 10^2 still **missed**, reported
coupon collector 5	0.140 s	n/a	12.65 s (12.64)[a]	**90**× **win**
Israeli–Jalfon 4	0.116 s	n/a	19.14 s (19.13)[a]	**166**× **win** (1048× vs. the published face, context only)
3-D asymmetric walk	3.94 s (u=200) / 1.84 s (u=150)	n/a	151.41 s (151.22)[a]	**38**× / **82**× **win**; both cells Pareto-dominate on both axes
Herman ring, 3 proc.	0.133 s (one pipeline, both queries)	54.65 s (54.44)	49.37 s (49.37)	**411**× / **371**× **win** — the one case both baselines run, both beaten end-to-end (932× / 466× vs. the published faces, context only)

[a] Diabolo's published timings: 47.93 s, 121.04 s, 289.94 s (its suite's slowest solved row); context only.

8.3 Tail asymptotics and termination mass

Diabolo automates exponential tail envelopes $O(c^n)$; the preregistered claim form was match-or-tighten c. All four expected-value cases **tighten**, three times at a rate that cannot be improved:

- coupon collector: house $5 \cdot (4/5)^n$ versus baseline $4.41 \times 10^9 \cdot 0.8002^n$ — the house rate *is* the chain's spectral radius $4/5$, so no smaller rate exists; coefficient 5 versus 4.4×10^9.
- Israeli–Jalfon: house $(99/64) \cdot (1/2)^n$ versus $4.1 \times 10^7 \cdot 0.5011^n$ — again the rate sits exactly on the spectral radius ($1/2 < 0.5011$), coefficient 1.55 versus 4.1×10^7.
- Herman ring: house $(3/4) \cdot (1/2)^n$ versus $5.6 \times 10^8 \cdot 0.50015^n$ same-box — the rate again sits exactly on the spectral radius ($1/2 < 0.50015$), coefficient 0.75 versus 5.6×10^8; and here the certified envelope is the exact tail's own leading term ($P[\mathit{count} > n] = (3/4)(1/2)^n - (1/8)(1/4)^n$), so beyond rate-optimality the bound's whole leading behavior is the truth's.
- 3-D walk: house $P[T > n] \leq 0.5617 \cdot 0.965^n$ — a *survival*-mass envelope, which dominates the baseline's density-form claim, at a tighter rate ($0.965 < 0.98148$ same-box) and coefficient (0.56 versus 396). The same certificate certifies the termination mass *exactly* 1 ($c < 1$ forces $P[T = \infty] = 0$), where the baseline's corresponding cell bounds it only within $[0.094,\ 20228]$.

8.4 The third axis, added rather than traded

Every number above ships as a replayable exact-rational container (0.7–20 KB), checked by an engine that shares no code with the producer; with a designed-locus mutant battery in class (roughly seventy-five negative loci plus positive controls across the nine cases, all verified through the committed recheck drivers); with 10^6-path source-semantics differentials including cross-program refusal teeth; and with exact ground-truth containment wherever a closed form or oracle exists. Neither baseline emits a witness or has an independent checker at any speed. The Pareto claim in §8.1 and §8.2 never leans on this axis — which is what makes it a Pareto claim.

8.5 Non-wins, collected

1. **The geometric-counter wall against GuBPI: 3.0×, against a preregistered 10^2×.** GuBPI is simply fast on this case (0.88 s); our end-to-end floor is ∼0.1–0.3 s, sealed lane included. Reported as a miss; no extenuation claimed.
2. **The Diabolo residual lane still outruns our checking path on the shared trio — the revision event settled this non-win against us, and the numbers say by how much.** Its residual-mass rows complete in 1–3 ms end-to-end; our pipeline takes ∼0.10–0.30 s. The first edition measured the staged closure re-validation at ≈92 ms of that total and predicted its removal by construction at the revision event. The event happened; the prediction held *where the tax dominated* (the token ring: −44% to −50% end-to-end) — but on the many-class trio the sealed checker's sharpened per-class reachability recompute costs approximately what the retirement saves (a same-load control in the receipts isolates the split), so the trio's end-to-end floor moved only marginally. The honest present tense stands: on millisecond-class cases, Diabolo without a witness is faster than us with one, and the table says so. What the event bought on these cases is not speed but grade: every trio verdict now rides the sealed revision, flag-free.
3. **The Beauquier tightness near-miss: CLOSED.** The first edition stood this row as a loss by 2.5×10^{-101}: GuBPI prints the exact point $[0.5, 0.5]$ — for a score-free program its normalizing constant is 1 by construction, and its float printout collapses to the truth — while our bracket carried the sound uniform residual above the exact lower endpoint 1/2. The reachability sharpening (§5.4) landed at the revision event: the bracket is now the point $[1/2, 1/2]$, width exactly 0 at both horizons, exact-rational, under strictly weaker hypotheses than the printout leans on (no termination assumption, no normalizing-constant argument). The same revision closed the Herman ring's protocol row (§8.1) to the width-zero point $[3/8, 3/8]$ before that case ever entered this list — the mechanism's second receipt, converted at birth. The row stays in this list as history — a loss that a rule revision converted, which is the calculus-growth mechanism working as designed. The wall on the same case remains under its preregistered 10^2 mark (60× at matched depth post-sealing, from 33×) and is reported as such.
4. **The 3-D flagship cell sits over its preregistered wall.** The horizon-200 cell (3.90 s) exceeds the preregistered 2.9 s mark, which the horizon-150 cell meets (1.79 s); both are reported, both Pareto-dominate the baseline, and the flagship's width is the one quoted against the prereg's tightness claim.

5. **Baseline-side honesty in the other direction.** Diabolo's published rows reproduced or improved on our box everywhere we ran it; its Israeli–Jalfon bracket was *confirmed sound* against the exact truth after our own preregistered hand derivation was falsified (§6.5); and its 3-D expected-value upper bound is optimizer-sensitive (same binary, same flags: 1.19×10^6 here, 6.85×10^5 published, each byte-stable on its own box) — quoted same-box, published face as context, per protocol.

9 Reproduction

The reproduction spine lives in the public `qtsl-experiments` repository, directory `probprog/`:

- `make probprog` — rebuilds the producer, re-derives every bracket, and regenerates the comparison table from committed receipts (`gen-table.ts`; the table in §8 re-derives byte-identically from `out/comparison.json`).
- `make probprog-glc` — re-checks every committed container through the fast-path checker built from the pinned `qtsl` checkout, and re-runs the container-lane mutant battery ("all verdicts as expected" is the green line).
- `make probprog-ev` — the expected-value lane: exact partial moments, contraction certificates, and the development-oracle battery.
- `make probprog-discrete` — the whole nine-case set, one verb.
- `make probprog-measure` — the timing harness (requires a load-quiet box and deliberate core pinning; numbers regenerate loudly, never silently).

Vendored beside the targets: the nine source programs (verbatim, MIT), the chain encodings and their documentation (`PROGRAMS.md`), the translation differentials, the raw measurement receipts (per-run stdouts, resource records, niceness read-backs, load averages), and both baselines' patch diffs. The containers, oracles, and Lean proof modules live in the `qtsl` repository beside the specification theories.

9.1 What separated [argued] from [shipped], resolved

The first edition named one step, the same for every row: the absorption species was *staged*, the contraction species' schema realization drafted, verdicts riding a flagged lane and a development oracle. The revision event has now executed, and its record discharged the first edition's own preregistered requirements in order: every container re-emitted against the sealed closure *byte-identically* (full-container comparison, twelve of twelve; the stop-the-line condition was armed and never fired — the sealed absorption closure's content-addressed revision is *the same revision* the staged lane checked against, because revision identity hashes semantic content and the pre-sealing motion was commentary plus checker semantics, so continuity is exact by construction); both mutant batteries re-run in class through the flag-free lane; the development-oracle lane retired for the finite-face expected-value rows, whose content now rides containers checked against the sealed contraction closure; and the staged-lane pin collapsed into the repository's standard theory pin — the reproduction make target runs with no workshop environment, and the formerly-staged invocation now *refuses*, so the flip is enforced rather than cosmetic. The wall table was re-measured, not extrapolated (§8.2).

What remains [**argued**]: the 3-D expected-value row. Its pattern face — finitely-presented infinite chains with product-profile templates — was deliberately deferred at the event's adversarial data-plane review: the face's schema still needs a state-coordinate binding, an explicit kernel species (a pattern's generated rows and a stored completion's rowless-is-absorbing reading are two different kernels and must not share one chain), and an infinite-carrier theorem bridging the finite-state induction to the pattern's state space. Sealing it without those would have minted a vocabulary two checkers could read differently, which is the one outcome this toolkit exists to make impossible. Its collapse mathematics is machine-checked and banked (at full profile generality — the per-class inequalities turn out to factor through a single worst class); its verdicts stay on the development oracle, labeled, until the face's own event. The deferral also produced a strengthening of the development oracle itself: a kernel-checked necessity witness showed the oracle's acceptance predicate under-demanded (a profile-length/window relation the collapse proof needs), and the oracle now enforces it — the shipped certificate satisfies it with room.

10 Related work and credit

This campaign exists because GuBPI and Diabolo built real, sound, public tools with real benchmark suites — the generous sentence about each is in §3 and is meant. Beyond the two baselines: PSI [3] supplies exact posteriors that serve as ground truth for the finite cases here (the burglarAlarm posterior 2969983/992160802 is PSI's number, re-derived by our checker from chain rows); the asymmetric-walk benchmark descends from work of Klinkenberg and colleagues [4]; the die paradox is Mossel's; exact-inference systems such as Polar and Prodigy appear in Diabolo's own evaluation as ground-truth providers rather than bound competitors, and we inherit that classification. The float-propose/exact-verify pattern inside Diabolo is the nearest methodological relative of our producer/checker split; the difference is architectural, not arithmetical — their verification lives inside the proposing process and leaves no artifact, ours crosses a process and format boundary that any third party can re-cross. A natural piece of future work in the generous direction: a witness-export patch for Diabolo, so that its own invariants could be checked by an external checker — ours or anyone's.

11 Summary

Nine benchmark cases from two published guaranteed-bounds tools — one of them run by both — same-box under preregistered claims. Tightness: win or tie on every shared case (three ties at the exact point — two of them width-zero conversions by one rule revision, on strictly weaker hypotheses than the printouts they tie; strict wins from exact-vs-float rounding through 3.0× structural to $9.2 \times 10^7\times$). Wall-clock: wins against every baseline running the case on six of the nine (2.2–411×, including the baseline suite's slowest row at 38–82× and the both-baselines case at 371× and 411×), GuBPI beaten by up to 713× on the shared trio and 60× on the token ring after the sealing halved that case's wall, the missed preregistered marks reported (§8.5), and one lane (Diabolo residual, millisecond-class) where the checking path — now sealed and flag-free — still loses outright, settled and said plainly. Tails: tightened on all four expected-value cases, three times at exactly the chain's spectral radius — once as the exact tail's own leading term; one termination mass certified exactly 1 where the baseline brackets it four orders of magnitude wide. Every number is an exact-rational, replayable, independently checked certificate with a mutant battery — an axis neither baseline occupies, added without being traded against either of theirs.

The durable output is the calculus growth the benchmarks forced: an absorption-bracket species whose truncation is sound by construction (a checked side condition dissolved into unrepresentability), hard conditioning as reject-classes with correlated-ratio posteriors whose histogram tightness is a single number, a contraction strengthening that subsumes its incumbent by machine-checked implication, a template family with a proven emptiness boundary, and an offset datum that turned a benchmark quirk into a general rule. The comparison table was the instrument; the calculus is the artifact.

Every absorption and finite-face expected-value row is [**shipped**]; the 3-D expected-value row is [**argued**], with the separating step stated in §9.1. The first edition's own re-survey triggers fired and are discharged in this edition: the species' revision event executed (the wall table re-measured, the residual-lane non-win settled against us and said so, the tightness near-miss converted). The triggers that remain, so this document keeps aging honestly: the pattern face's own revision event (which flips the last [**argued**] row or reports why not); any extension to the baselines' continuous or recursive suites, which this report deliberately does not claim; and any third-party reproduction from the public make targets, which is the point of having them.

References

[1] R. Beutner, C.-H. L. Ong, F. Zaiser. *Guaranteed Bounds for Posterior Inference in Universal Probabilistic Programming*. PLDI 2022. arXiv:2204.02948. Artifact: github.com/gubpi-tool/gubpi (MIT).

[2] F. Zaiser, A. S. Murawski, C.-H. L. Ong. *Guaranteed Bounds on Posterior Distributions of Discrete Probabilistic Programs with Loops*. POPL 2025. arXiv:2411.10393. Artifact: github.com/fzaiser/diabolo (MIT); Zenodo 10.5281/zenodo.14169507.

[3] T. Gehr, S. Misailovic, M. Vechev. *PSI: Exact Symbolic Inference for Probabilistic Programs*. CAV 2016.

[4] L. Klinkenberg, K. Batz, B. L. Kaminski, J.-P. Katoen, J. Moerman, T. Noll. *Generating Functions for Probabilistic Programs*. LOPSTR 2020.

[5] David "davidad" Dalrymple and the maitria coalition contributors. *MAITRIA: The Book*. Living draft, CC BY 4.0; `maitria.org/book.pdf`, with its full source embedded in the PDF as an attachment.

Chapters are cited by title; the coverage dashboard of §4.2 is the closing section (*Distribution generators: a coverage dashboard*) of the chapter *Hypernets: Markov Kernels*.

[6] G. E. Crooks. *Field Guide to Continuous Probability Distributions*. 2022. Available at `threeplusone.com/pubs/FieldGuide.pdf`.

Source appendix. The following pages carry the complete LaTeX source of this document (`probprog-bounds.tex`), on one or more consecutive pages per source file (large files are split across pages at line boundaries), as an invisible text layer: extract it with `pdftotext -layout <thisfile>.pdf -` (the source begins at the first `%` comment or `\documentclass` on the next page; the layer never prints or displays). That recovers every line exactly, except that interior runs of exactly two spaces collapse to one; `pdftotext -layout -fixed 1.3226 <thisfile>.pdf -` is byte-exact throughout. The same files are also embedded byte-exactly as PDF attachments: `pdfdetach -saveall <thisfile>.pdf`, or `reader.attachments` in pypdf.