A computational supplement to the encounter

thinksophia

Bounded Encounters and Recoverable Work

Johan Newaz · Founder and blueprint author
AI-assisted technical supplement · 21 October 2026 · First bounded iteration

This supplement gives the previous companion a more exact mathematical footing. It develops three connected pieces: a dispatch boundary that respects current authority, a resource ledger that does not spend the same capacity twice, and a finite ripple that starts and settles smoothly. The result is a small executable reference model, improved figures and a reviewable account of what the checks establish.

The main finding is conditional but concrete. In the specified two-worker model, all 1,120 allowed event orders preserved the budget and dispatch rules across 8,960 transitions. Eight focused checks also passed. Two deliberately flawed accounting schedules exceed the budget. These are results from a synthetic model, not tests of the deployed thinksophia site or observations of visitors.

The next sip

Johan asked for the figures, diagrams, mathematics and computational metrics to mature while he reads the richer companion. This is a separate document, not a replacement. The original fourteen-page companion and the two earlier sets of three documents remain unchanged. “Step, sip, test, recover” remains the organizing rhythm, not a claim about how every human encounter must unfold.

Current academic work informs the questions and proposed safeguards. It does not validate this design by association. A literature section distinguishes accepted research, position work, preprints and established distributed-systems foundations, with dates and direct sources.

How to read it

Read the model and worked trace for the operational idea; read the proof and numerical section for the mathematics; read the results and limitations before interpreting the figures as evidence. The code and machine-readable results accompany the document so the bounded claims can be reproduced.

Separate the visitor from the operation

The first companion used named states to make an encounter legible. A mature model needs more than one axis. The visitor may leave while an authorized operation is in flight; an external effect may be known while its final charge remains unknown. One word such as closed or failed cannot faithfully describe all those facts.

$$x = (m, G, J, U, R, B, H)$$

Here m is the interaction mode; G the current grants; J the operation records; U settled resource use; R outstanding reservations; B the resource limits; and H the relevant dispatch and outcome history. This is operational state, not a psychological model. Attention is represented only by explicit interface events where needed, never by an inferred measure of sincerity or inner experience.

Object	Domain	Meaning
m	ready, paused, closed	Interface availability and continuation policy
G	scoped grants	Action, data, destination, purpose and validity conditions
J[j]	operation record	Immutable ID, phase, evidence, cancellation and reservation
U_i, R_i, B_i	nonnegative resource units	Settled use, exposure still reserved and limit
H	ordered event records	Minimal evidence at enforcement and reconciliation points

Each operation should distinguish its execution phase, evidence and cancellation status. Phase can be reserved, in flight or terminal. Evidence can be pending, uncertain, confirmed effect or confirmed no effect. Cancellation can be absent, requested, acknowledged or confirmed. Acknowledgment is not proof that the destination prevented an effect.

A deliberately smaller executable model

The accompanying code collapses these axes into six operation labels: reserved, in_flight, uncertain, committed, failed and cancelled. It models one Boolean grant and one stop flag, two operation IDs and one integer resource. This smaller abstraction makes complete enumeration possible. It does not claim to implement the richer product state, account authentication, networking, storage or a policy language.

Authority belongs at the dispatch boundary

Define the enforcement point as the instant an operation is admitted to outbound dispatch in a serialized controller. At that point the relevant grant must be valid, stop must not block the operation, policy must permit it and the operation must hold sufficient reserved capacity. A real design must identify this point in code and document which authority service controls it.

$$\text{admit_dispatch}(j, t) = \text{scope_valid}(j, G(t)) \wedge \text{policy_ok}(j) \wedge \text{not_stopped}(j, t) \wedge \text{reservation_valid}(j)$$

The important invariant concerns history: every admitted dispatch had valid authority at its enforcement point. It is not a requirement that a grant remain valid forever after a legitimate action. Later expiry or revocation must not retroactively turn a correct historical record into a violation.

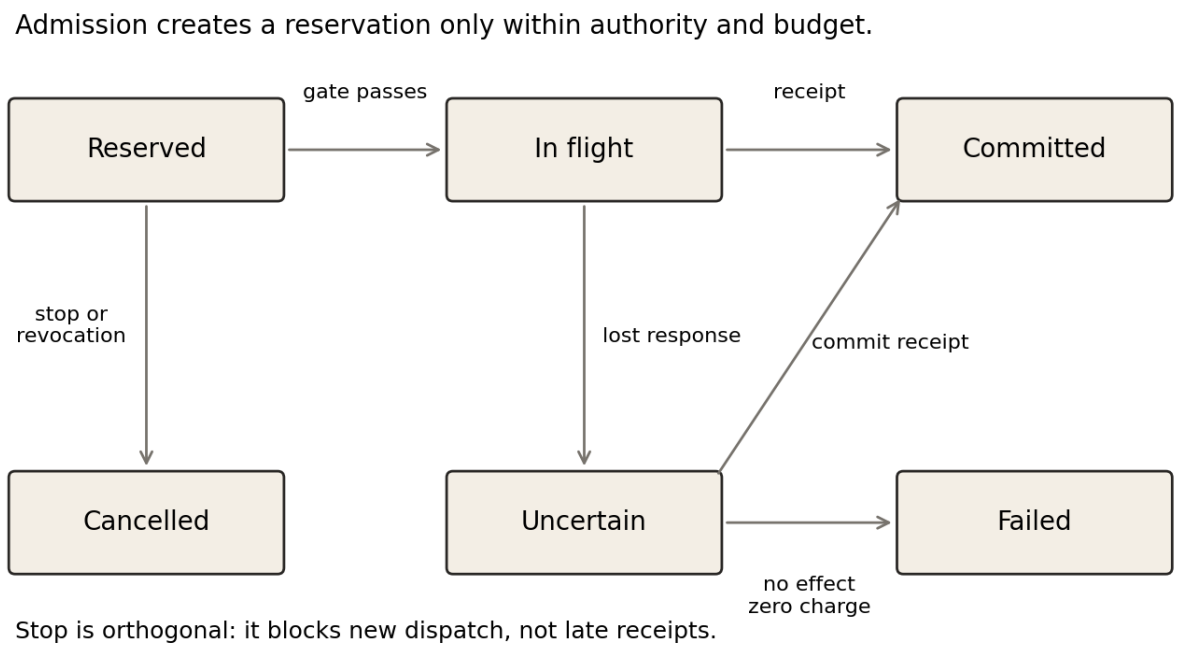


Figure 1. Proposed operation paths. The diagram is a design model, not a deployment trace. Admission and stop are serialized in the executable abstraction.

A local gate can guarantee no new dispatch passes after effective local revocation. It cannot by itself guarantee that a previously dispatched request will not produce a remote effect later. That stronger property needs destination cooperation, such as a destination-side grant check or supported cancellation protocol. “No effect after stop” is therefore not the result claimed here.

A refusal addresses conduct and scope, not a person’s worth. The cactus remains a useful image for protection at a boundary, not pursuit beyond it. Reconciliation, compensation and disclosure each need their own authority; a disputed event is not blanket permission to investigate or expose someone.

A ledger that cannot lend the same cup twice

For each cumulative resource i, define B_i as the fixed limit, U_i as settled consumption and R_i as the sum of outstanding reservations w_{ji}. Units must match within a dimension. Tokens, tool calls, seconds of execution and currency remain separate dimensions; summing them without a declared conversion model is meaningless.

$R_i = \Sigma_j w_{ji}$ and $U_i \geq 0, R_i \geq 0, U_i + R_i \leq B_i$

Admission

At one atomic ordering point, admit a reservation only when every resource dimension fits. Then increase R by the reserved vector. Concurrent workers must not independently read the same free capacity and both promise it. “Atomic” here means the check and update appear indivisible to competing operations; it is an implementation requirement, not something a prose instruction to an agent supplies.

If $U_i + R_i + w_i \leq B_i$ for every i, then $R_i' = R_i + w_i$

Settlement

On a definitive, deduplicated receipt with actual consumption c_i bounded by the reservation, move consumption into U and release that operation’s reservation. A failed effect may still incur cost. The code’s fail event is deliberately narrower: a confirmed no-effect receipt with confirmed zero charge.

$U_i' = U_i + c_i$ $R_i' = R_i - w_i$ with $0 \leq c_i \leq w_i$

The accounted total changes by $c_i - w_i \leq 0$, so settlement preserves the bound. Admission preserves it by its guard. Starting from a valid ledger, any sequence of these two transitions preserves the invariant, assuming atomicity, correct receipts, bounded consumption and no other transition violates it. This is an inductive argument about the specified rules, not a proof that an external provider obeys a cost cap.

Do not reduce B below current U + R. Partial bills need a transfer from reservation to settled consumption, not double counting. Simultaneous worker capacity is not cumulative expenditure and needs a separate occupancy model. Unknown fees, conversion rates or unbounded tool costs invalidate the claimed hard bound unless controlled or conservatively covered.

Uncertainty keeps its place in the ledger

A lost response leaves at least two possible histories: the external operation committed, or it did not. Neither becomes true merely because the interface prefers a clean ending. A timeout changes knowledge, not the past. The corresponding reservation remains held until an authoritative receipt or another justified reconciliation rule resolves the exposure.

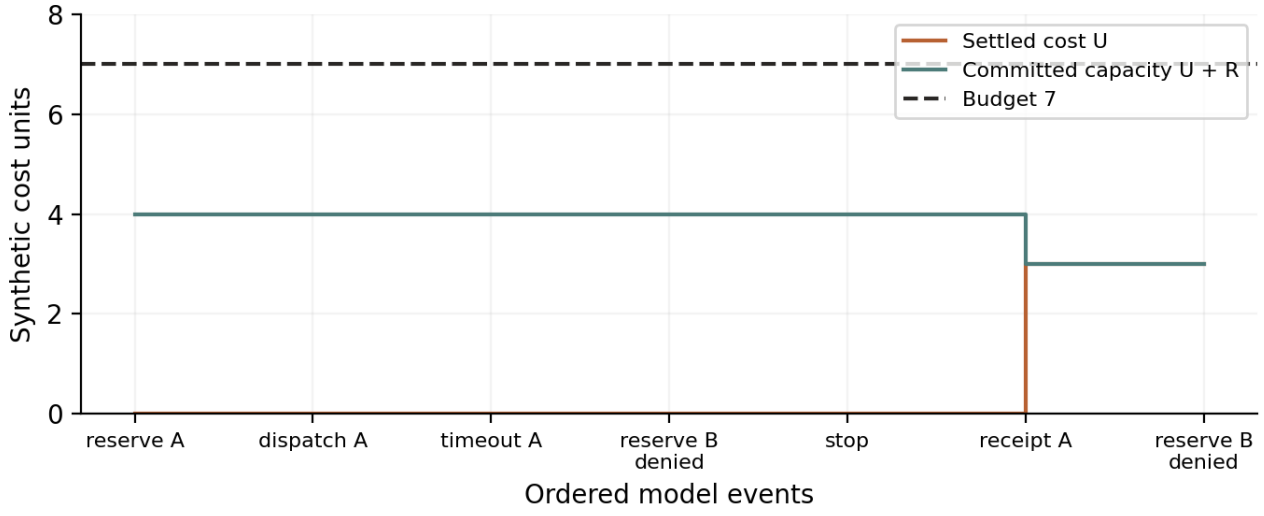


Figure 2. Executed synthetic trace from check_model.py. Cost units are invented. The timeout holds capacity; the stop blocks later admission; a late receipt settles an already dispatched operation.

In this trace the budget is seven units. A reserves four, dispatches and becomes uncertain. B cannot reserve another four. After a stop, A's receipt confirms a three-unit charge, leaving settled use at three and reservations at zero. B remains blocked by stop even though capacity is available. Capacity is necessary for admission, not sufficient authority.

Recovery has a cost

A real status check may consume time or money. Its scope and reservation must be available separately. The executable model treats receipts as incoming events with no modeled cost, so it does not demonstrate a funded reconciliation service. A future version should preallocate reconciliation capacity or explain how a request for additional authority works when capacity is exhausted.

Conservative reservation protects against overspending but can prevent progress indefinitely if evidence never arrives. Safety does not imply liveness. Never erase unknown liability merely to make the system appear responsive. Expose the unresolved status and a proportionate next option.

The gardener's care remains distinct from the investigation of blame. In operational terms, offer safe local help now, disclose uncertainty and avoid an irreversible repair or repeated external action until its authority and consequences are clear.

What the computation actually checked

The reference program uses two workers, A and B. Each has reserve, dispatch and settle events in that order. A revoke event and a stop event can occur anywhere. There are eight events total. Enumerating all total orders that preserve each worker's three-event order yields 8! / (3! × 3!) = 1,120 schedules.

Each schedule begins with budget seven, zero settled use and zero reservations, a valid grant and no stop. Each reservation requests four units; each successful settlement costs three. Events whose prerequisites are absent are no-ops. For example, dispatch after denied admission does nothing. After every event, assertions check nonnegative reservations, U + R ≤ B and the recorded authority and stop state of every dispatch.

Check	Observed result	Scope
Schedule enumeration	1,120 schedules; 8,960 transitions	All specified serialized event orders
Dispatch records	432 admissions across schedules	Each passed the Boolean grant and stop gate
Invariant violations	0 in these schedules	Not a statistical failure-rate estimate
Focused cases	8 passed	Specific uncertainty and cancellation scenarios
Negative controls	2 expected violations shown	Stale reads and premature release

This is exhaustive over this finite abstraction, not over all implementations. It assumes a serializable gate, exact integer accounting and fixed charges. It does not run real parallel threads, model remote services or explore arbitrary numbers of workers, mutable grants, clock failures, crashes, billing streams or retries. A larger model can violate an invariant through transitions absent here.

Why the negative controls matter

In the stale-read schedule, A and B both see seven free units, then each reserves four without a shared atomic update. The result is eight reserved units against seven. In the premature-release schedule, A's unresolved four-unit liability is dropped and B reserves four; if A later incurs four, exposure again reaches eight. The included arithmetic controls assert these violations. They are constructed counterexamples, not an empirical comparison of database systems.

Focused checks and their boundaries

Case	Executable assertion	What remains outside the check
Uncertain cost	Timeout retains four units and blocks B	Provider costs and eventual receipts
Repeated receipt	A second settle does not increase U	Out-of-order partial billing
Stop and late result	Dispatched A can settle after stop	Actual destination cancellation
No blind retry	Dispatch cannot repeat uncertain A	New IDs or expired external keys
Revocation	Reserved A cannot dispatch after revoke	Distributed grant caches
Definitive failure	No-effect zero-charge receipt releases R	Failures that incur charges
Cancel before dispatch	Undispatched reservation is released	Crash recovery of persisted reservations
Cost cap assumption	Synthetic cost three fits reserve four	Provider overruns or hidden fees

These eight checks execute assertions against the reference class. They are a first layer, not a substitute for tests of the missing mechanisms. Repeated receipts are handled because a terminal job cannot settle again. That does not establish exactly-once behavior at an external destination. A new operation ID, a lost checkpoint or an expired provider deduplication window can change the situation.

The next failures worth injecting

A later bounded iteration should add crash-after-reserve recovery, partial settlement, authoritative versus eventually consistent lookups, a receipt that exceeds its reserved cap, checkpoint expiry and a reconciliation budget. Each extension should bring a named transition, its assumptions and a counterexample that would falsify the intended rule.

The original eight proposed tests remain proposals for the full interaction system. This supplement executes narrower model checks related to authority, shared budget, uncertainty and stopping. It does not execute accessibility, real retention deletion, authentication or human welcome tests.

A finite ripple with a smooth return

The original illustrative ripple used an exponentially damped sinusoid with a hard cutoff. With A = 1, τ = 0.65 seconds, f = 1.5 hertz and T = 2.5 seconds, its left-hand endpoint value is approximately −0.02136, so setting it immediately to zero creates a small discontinuity. Its start value is zero, but its right derivative is about 9.42478 per second against a zero derivative before onset.

$$r(t) = A \exp(-t/\tau) \sin(2\pi f t) \sin^2(\pi t/T), \quad 0 \le t \le T$$
$$r(t) = 0 \text{ otherwise}, \quad T > 0 \text{ and } \tau > 0$$

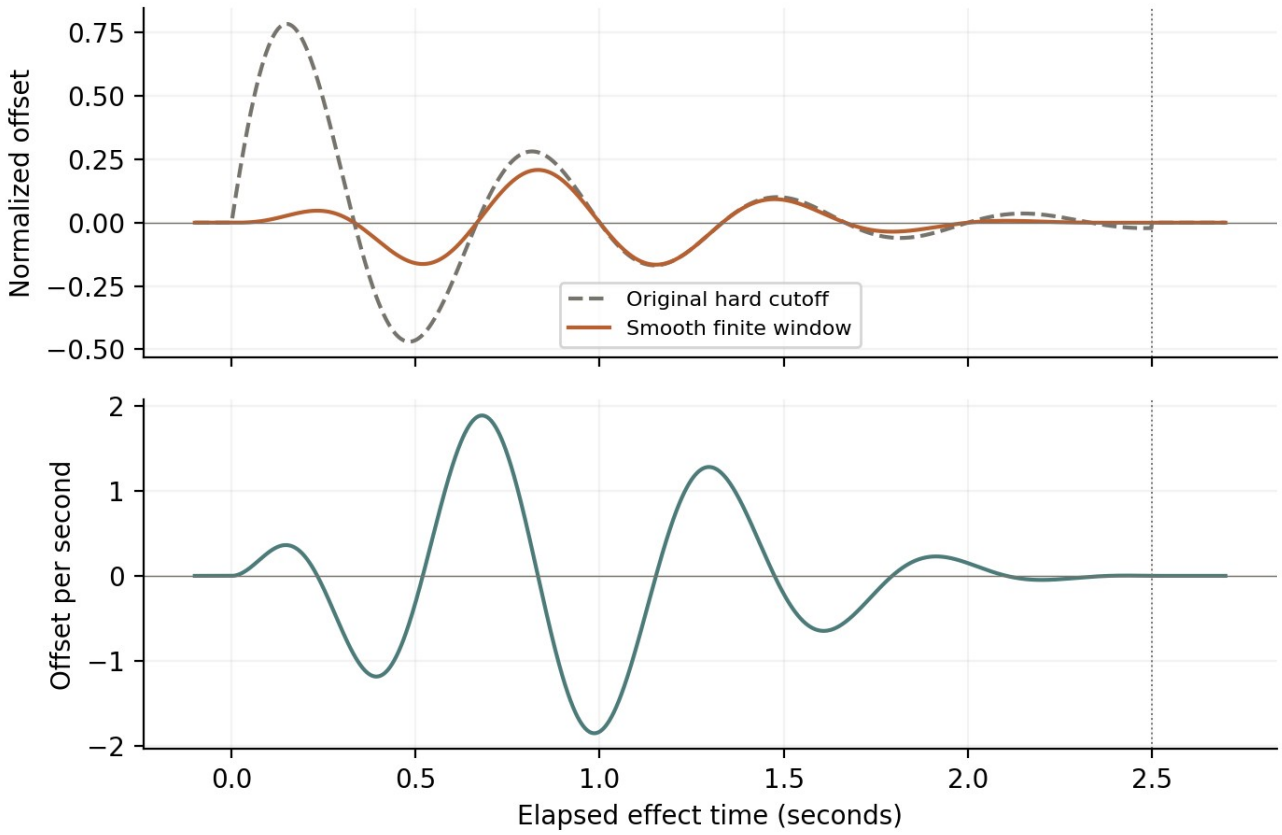


Figure 3. Computed analytic curves, not observed motion. Top compares the original cutoff to the finite window; bottom shows the derivative of the new curve. Parameters are illustrative, not comfort recommendations.

The $\sin^2$ window is zero with zero first derivative at both ends. The zero-extended function is therefore C1: position and first derivative join continuously to rest. This is a different window from the cosine-squared terminal taper suggested in the first companion, chosen here because it also smooths the onset. It reduces the attained peak and changes the motion; that is a design trade-off, not a free correction.

Derivation and numerical checks

$$\omega = 2\pi f, \quad \beta = \pi/T$$
$$r'(t) = A \exp(-t/\tau)\,[(\omega \cos(\omega t) - \sin(\omega t)/\tau)\sin^2(\beta t) + \beta \sin(\omega t)\sin(2\beta t)]$$

Inside the interval this follows by the product rule. At zero and T, the window and its derivative vanish, so r and r′ vanish. For arbitrary parameters the zero extension is not C2 at T: $r''(T-) = 2A \exp(-T/\tau)\sin(\omega T)\beta^2$, which need not be zero. The model should claim C1, not unlimited smoothness.

t, τ and T are measured in seconds; f in cycles per second. A and r are normalized displacement here, so r′ is normalized displacement per second. A is a scale and upper bound, not the attained peak. For $A \geq 0$, $|r(t)| \leq A \exp(-t/\tau) \leq A$. Physical pixels could be introduced by an explicit scale conversion, not by treating the normalized chart as a fluid experiment.

Computed quantity	Result	Interpretation
Sample grid	10,001 points on [0, 2.5] seconds	Numerical sampling, not the analytic proof
Sampled maximum |r|	0.2081091	Below scale A = 1
Derivative comparison	Maximum error about 3.33×10^{-9}	500 interior points, central step 10^{-5} seconds
Endpoint difference quotients	Approach zero as h decreases	Consistent with the analytic C1 result
Outside support	Exactly zero in implementation	Defined by the piecewise function

The derivative comparison checks the coded formula against a numerical approximation. It does not prove a general theorem; the endpoint and bound arguments do that for the mathematical expression under their assumptions. Finite precision, frame sampling, graphics transforms and device scheduling still matter in an implementation.

Keep one active response rather than adding arbitrary copies: superposition can exceed a single-response amplitude bound. A retrigger rule should either ignore input until completion or replace the effect with a separately designed continuous transition. C1 motion does not establish accessibility, delight or absence of discomfort. A quiet and reduced-motion path remains necessary to evaluate separately.

Metrics that do not confuse proof with a score

Report technical observations with explicit denominators, windows and failure semantics. The enumeration's zero violations is a statement about a finite model and fixed schedule set. Dividing zero by 8,960 does not turn it into a real-world risk estimate. The number 432 counts admissions across constructed schedules, not independent users or trials.

Metric	Definition	Required context
Dispatch violations	Invalid admissions / all admitted dispatches	Authority evaluated at the stated enforcement point
Duplicate external effects	Excess committed effects / logical operations retried	Destination evidence and stable operation identity
Stop-to-quiescence	First verified closed-gate time at or after accepted stop, minus stop time	Nonnegative; unresolved if closure is not verified
Unresolved exposure	Σ outstanding reservations by resource	Do not sum incompatible units
Reconciliation delay	Verified outcome time minus uncertain classification time	Keep unresolved cases censored and visible
Task outcome mix	Completed, failed, partial and uncertain counts	Fixed eligibility and observation window

A denominator of zero means the rate is undefined. Report counts with rates. For reconciliation, do not drop unresolved cases from the summary: show their number, ages and reserved exposure. A task that ended cleanly is different from one that merely disappeared from the observation window.

Gate quiescence means all relevant local admission gates are verified closed to new work, not that remote effects have stopped. Already closed at acceptance means zero delay. If closure is unverified, report a censored interval, never a negative delay. Separately count dispatches admitted after accepted stop. Use a monotonic clock for local elapsed intervals. Distributed timestamps need a synchronization error bound; otherwise report a partial order or interval rather than pretending millisecond precision. The synthetic program uses event order, not elapsed wall time, so it has no measured cancellation latency or throughput result.

Keep human meaning voluntary

A visitor may describe usefulness, clarity or afterglow in an optional question. Cursor movement and time on page do not supply those answers. The computational metric here concerns observable work and resource commitments. It does not grade the visitor, infer intent from silence or give a numerical definition to the artistic imagery.

Current research on authority and reliability

The following is a targeted primary-source review checked on 21 October 2026, not a systematic literature survey. Sources are selected for concrete implications for the model, not for novelty alone. Their empirical results do not transfer to thinksophia without reproducing the relevant conditions.

Authenticated delegation

Tobin South and colleagues, Position AI Agents Need Authenticated Delegation, ICML 2025, PMLR 267, 13–19 July 2025. Peer-reviewed position paper. The authors argue for authenticated, auditable delegation and discuss scoped authorization using established identity protocols. This is an architectural argument, not an empirical assurance study.

Implication: keep authority distinct from task intent. A fuller operation record should identify principal, delegate, action, target, purpose, expiry and policy version. Test target substitution and subdelegation that exceeds the original grant. Authentication alone does not establish that natural-language intent was translated correctly.

Primary source: https://proceedings.mlr.press/v267/south25a.html

A profile rather than one reliability score

Stephan Rabanser and colleagues, Towards a Science of AI Agent Reliability, ICML 2026, PMLR 306, 6–11 July 2026. Peer-reviewed conference paper; v3 preprint dated 2 June 2026. The paper organizes twelve metrics into consistency, robustness, predictability and safety. Its fifteen-model analysis on GAIA and tau-bench reports reliability gains that lag capability gains.

Implication: the next evaluation should vary equivalent instructions, tool faults and repeated runs while holding budgets constant. Report severe failures separately. The present deterministic schedule explorer evaluates controller rules, not language-model reliability. Two benchmarks, particular scaffolds, temperature-zero runs and model-assisted safety judgments limit generalization.

Primary sources: https://proceedings.mlr.press/v306/rabanser26a.html
https://arxiv.org/html/2602.16666v3

Runtime enforcement and cost aware evaluation

A recent formal authorization preprint

Yifan Zhang and colleagues, FAVA Formal Authorization for Verified Agents with Evidence Backed Permission Graphs, arXiv:2607.27267v1, 29 July 2026. Preprint; no peer-reviewed venue verified. FAVA combines a permission representation, evidence and data-flow graphs, solver-backed authorization and runtime mediation. It reports 90.5% decision compliance on 801 replay cases, alongside a 57.7% benign allow rate.

That trade-off matters: blocking legitimate work can make a host unusable even while a decision metric looks strong. The abstraction depends on complete mediation and trusted translation of policy and evidence. This supplement does not implement FAVA or inherit its guarantees. A future test should measure false blocks as well as unauthorized permits and deliberately probe hidden data-flow paths.

Primary sources: https://arxiv.org/abs/2607.27267
https://arxiv.org/html/2607.27267v1

Capability is not independent of spending

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir and Arvind Narayanan, AI Agents That Matter, Transactions on Machine Learning Research, May 2025; initial preprint 1 July 2024. Peer-reviewed journal paper. It argues that accuracy-only comparisons confound capability with resource expenditure, and motivates cost-controlled, reproducible evaluation and accuracy–cost frontiers.

Implication: compare candidate controllers at matched budgets, retain unsuccessful attempts in cost accounting, and distinguish tokens, calls, retries, elapsed time and human intervention. The reservation invariant here is our design inference, not a theorem from the paper. Its task-specific experiments do not promise a particular saving for this encounter system.

Primary record: https://openreview.net/forum?id=Zy4uFzMviZ
Author preprint: https://arxiv.org/html/2407.01502v1

The current computational results test exact synthetic accounting. They contain no vendor prices, token-cost measurements or Pareto comparison. Introducing those quantities would require a dated price model and actual comparable runs.

Older systems results still define the hard edges

Remote results can remain unknown

Andrew D. Birrell and Bruce Jay Nelson, Implementing Remote Procedure Calls, ACM Transactions on Computer Systems 2(1), February 1984, pages 39–59. Peer-reviewed foundational systems paper. Its protocol distinguishes a successful return from communication failure that can leave the caller unable to determine whether the invocation occurred; call identifiers address duplicate delivery.

Implication: keep timeout separate from confirmed no effect. Correlate authoritative receipts with a stable operation ID before repeating a consequential request. The paper describes a particular protocol, not a universal exactly-once guarantee for arbitrary modern tools. Our synthetic timeout case represents uncertainty but does not implement an RPC transport.

Primary author copy: https://birrell.org/andrew/papers/ImplementingRPC.pdf
DOI: https://doi.org/10.1145/2080.357392

Compensation is another action

Hector Garcia-Molina and Kenneth Salem, Sagas, ACM SIGMOD 1987, pages 249–259. Peer-reviewed foundational database paper. It decomposes long-lived transactions into steps with compensating transactions and recovery machinery. Compensation is application-defined; it is not an eraser of history.

Implication: an attempted repair needs an explicit operation, authority, budget, outcome and failure treatment. A cancelled local job may leave a committed remote effect, and a compensator may itself fail or become uncertain. The current model has no compensation transition. Adding one must not silently reclassify an irreversible effect as absent.

Primary institutional report: https://www.cs.princeton.edu/techreports/1987/070.pdf
Proceedings DOI: https://doi.org/10.1145/38713.38742

A useful synthesis

Keep five things distinct: intent, authority, knowledge of effects, resource exposure and recovery obligations. Current agent research makes these distinctions urgent; older distributed-systems work explains why they cannot be solved merely by a more confident response. The choice to implement them remains a separate engineering decision.

Reproduce the next sip

The companion package includes check_model.py, results.json, run.log and the three figure images. The Python program has no network calls, credentials, real users, live payments or external side effects. It regenerates its result JSON and figures from fixed synthetic inputs. It requires Python 3, NumPy and Matplotlib; the recorded run used Python 3.12.14. Dependency versions are recorded in the verification manifest.

```
python check_model.py
```

Run without Python optimization flags because the checks use assertions. Success means the process exits normally and writes results.json; failure raises an assertion or another exception. Compare the counts and numerical results to the document. Small floating-point differences can depend on platform libraries; a changed plot or test count should be explained, not silently accepted.

What changed from the first companion

Authority is now explicitly historical at the dispatch boundary. Stop is separate from late outcome evidence. Reservations have an inductive invariant and a stated upper-bound assumption. The finite effect has a derived C1 envelope and a plotted comparison. The model has executable evidence rather than only proposed tests. Academic sources motivate additional failure cases without claiming that their findings transfer automatically.

What has not changed

The existing documents and website remain intact. No production harness, public release, remote cancellation integration, human experiment or general intelligence claim is established here. The small program is a reproducible reference calculation and schedule explorer. It is not a security boundary for real actions.

A useful next implementation decision would select one of the uncovered failures, such as persisted reservations after a crash, and extend the model with a narrow test. That decision remains separate from this delivered supplement. The work can become more exact without asking the visitor to consume every page at once.

The cup is finite. What it contains should be clear.