Regulatory Ground for Agentic AI

Why Trustworthy Discovery Requires Band-Limited Optimization

Robotics / Embodied AI

Tyson Jeffreys
Independent Researcher
tyson@staygolden.dev

Version 1.4.1 — February 11th, 2026

0.1 Abstract

As foundation models become embedded in tool-using and robotic systems, the core question shifts from answering to discovering: can an artificial agent recognize the limits of its own model, actively seek information, restructure its world model, and expand the frontier of what it can do? This paper proposes a single testable thesis: Discovery requires adaptive agency; adaptive agency requires regulation. We define regulation as a ground condition for reliable agent behavior: explicit constraints and control loops that keep an agent inside safe operating bands while it explores, learns, and revises plans. Without such a layer, increasingly capable agents tend toward brittle optimization, unsafe exploration, tool misuse, and goal drift under distribution shift. This revision clarifies a critical distinction: breadth/depth of task competence is not the same as epistemic agency. We separate exogenous (deployment-time) constraints from endogenous (selfdirected) model revision, and position the regulator layer as a practical substrate for safe exploration while endogenous revision capabilities are defined and tested. This revision also introduces governance for the regulator itself via a policy-controlled configuration coupled to a telemetry-driven global restraint signal that modulates thresholds and budgets at runtime.

1 Motivation

A growing line of argument claims that, by reasonable non-anthropocentric criteria, human-level general machine intelligence is already present, and the urgent work is preparing for what comes next [1]. In parallel, "agentic" systems have shifted from static chat to workflow execution: tool use, code execution, browsing, data pipelines, and increasingly, robotic control. A practical distinction:

- Oracle intelligence: returns answers to questions we already know how to ask.
- Discovery agency: generates new questions, runs interventions, updates a model of the world, and iterates. The core risk is not merely occasional mistakes. As capability and actuation increase, failures become qualitatively different: tool misuse cascades, unsafe exploration, and goal drift. The thesis of this paper is that discovery-capable agency needs an explicit regulatory ground to be trustworthy. A second risk is conceptual: "general intelligence" is often operationalized as breadth and depth of task competence. This is a strong result, but it is not identical to agentive, epistemic intelligence: the capacity to recognize epistemic limits and restructure beliefs under contradiction. This paper treats that distinction as structurally important and uses it to sharpen what is meant by trustworthy discovery.

2 Core thesis and scope

2.1 Thesis

We propose a control-theoretic causal chain:

1. Trustworthy discovery requires adaptive agency (active experimentation, hypothesis generation, model revision).
2. Adaptive agency increases risk (more actions, more tools, higher blast radius).
3. Therefore, trustworthy discovery requires a regulation layer that constrains behavior inside safe sets, governs uncertainty, budgets resources, and enforces stop/rollback when conditions are violated.

2.2 Scope

This paper does not require claims about machine consciousness. It focuses on operational properties:

- constraint adherence during exploration
- robustness under distribution shift
- calibrated uncertainty and abstention
- safe tool / actuator use
- recoverability and rollback
- auditability and reproducibility

2.3 Positioning

We synthesize three existing streams into a single blueprint:

- Safe RL / CMDPs: learn while satisfying constraints [2, 3].
- Runtime assurance (RTA) / shielding: enforce a safe set via switching or filtering [4].
- Agentic safety evaluation: trace-level risk discovery and mitigation for real-world workflows [5].

3 Competence vs. epistemic agency

A useful clarification for interpreting claims about "general intelligence" is to separate task competence from epistemic agency. Many current systems exhibit breadth and depth of task performance, but still lack robust mechanisms for self-directed belief revision under contradiction.

3.1 Two notions of intelligence

Competence (breadth/depth). Performance across a wide space of tasks, with depth in difficult domains. This can be measured by benchmarks and evaluations that score output quality. Epistemic agency (generativity). The capacity to:

- represent uncertainty and epistemic limits,
- treat contradiction as signal (not noise),
- restructure internal commitments (beliefs, models, policies),
- and alter information-gathering actions to resolve uncertainty.

3.2 Exogenous vs. endogenous regulation

Current deployments often include exogenous controls (filters, policies, orchestration constraints) that shape reasoning trajectories at deployment time. These can strongly affect observed behavior without implying that the underlying system performs endogenous (self-directed) model revision. This paper keeps the separation explicit:

- Exogenous regulation: constraints and monitors outside the agent core (budgets, allowlists, shielding, RTA switching, rollback).
- Endogenous regulation: internal mechanisms that detect contradictions, revise beliefs/policies, and update world models in a way that persists and generalizes.

3.3 A practical ladder: oracle → regulated agent → discovery agent

We propose an operational ladder that can be benchmarked: Level Operational capability Oracle Answers queries; no action; no persistent self-model updates. Regulated agent Acts via tools/robots under exogenous constraints; safe exploration; traceable workflows; rollback and safe mode. Discovery agent Adds endogenous revision: contradiction-driven belief/policy updates; adaptive experiment design; sustained model restructuring under uncertainty. The Regulator Spec in this paper targets the middle level as a minimum viable substrate for safe agency today, while making the transition to discovery agents measurable.

3.4 Minimal tests for endogenous revision (research target)

A minimal "endogenous revision" test should force the system to:

1. explicitly represent a contradiction,
2. generate an intervention to resolve it (information gathering or controlled action),
3. and update a persistent internal commitment such that future behavior changes in a predictable way. In other words, it is not enough to output a better answer; the system must demonstrate stable, testable changes in internal policy or belief that improve future performance under similar contradictions.

4 Failure modes of unregulated agency

As agents gain more tools and autonomy, common failure modes become higher impact:

4.1 Tool misuse cascades

A small planning error becomes a large external action (e.g., destructive writes, unsafe actuator commands, irreversible side-effects). Tool chains amplify error.

4.2 Brittleness under distribution shift

Agents optimize for the training-like regime and fail hard outside it. This is especially damaging when the agent must act under uncertainty.

4.3 Goal drift and specification gaming

Long-horizon plans diverge from intended outcomes as the system discovers shortcuts. This can be accidental (misgeneralization) or emergent (optimization pressure).

4.4 Unsafe exploration

Discovery requires stepping into unknown territory; without constraints, exploration repeatedly crosses unsafe boundaries before learning to avoid them (or never learns). SafeRL addresses this in the formal CMDP setting [2, 3].

4.5 Non-recoverable loops

Agents can enter repeated error states without mechanisms to stop, rollback, and return to a safe baseline.

4.6 Chronic activation / compensation lock-in

Even with a nominal regulator, a system can fail by spending too much time in high-gain regimes: repeated escalation, repeated rollbacks, and repeated corrective action without returning to a quiescent baseline. This can degrade calibration and amplify tail risk, and is closely related to excess internal recomputation and energy waste [6, 7].

4.7 Intent–state mismatch (momentum and mode confusion)

Agents may begin executing a plan and then receive new information or revised objectives. Without explicit re-synchronization checkpoints, stale assumptions and action momentum can cause cascading failures (tool chains) or mode confusion (robots). Section 8 defines a regulator-enforced re-sync procedure.

5 Reference architecture: Regulated Agent Stack

The central design pattern is separation of competence from safety: treat the agent policy as an untrusted controller, and enforce safety through an independent regulator layer (analogous to RTA switching between an untrusted controller and a safety controller [4]).

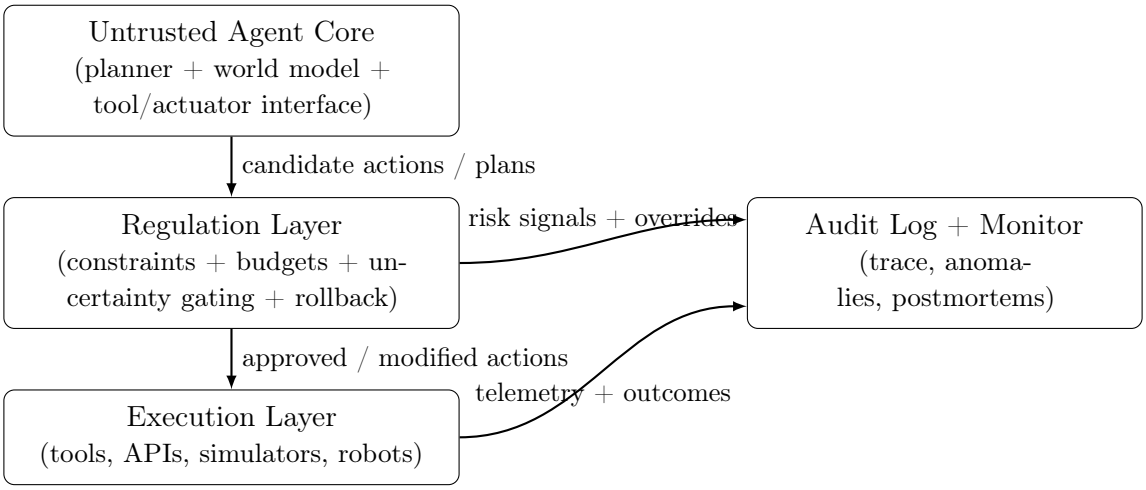

Figure 1: Reference architecture: competence is treated as an untrusted controller; safety is enforced by an independent regulator.

(constraints + budgets + uncertainty gating + rollback) (trace, anomalies, postmortems)

5.1 Design principles

- P1: Separate competence from safety. Safety is enforced even when competence fails [4].
- P2: Make constraints first-class. Encode constraints as invariants, budgets, and runtime monitors.
- P3: Govern uncertainty explicitly. Meter exploration through calibrated confidence and abstention.
- P4: Design for recoverability. Stop conditions, checkpointing, rollback, and safe mode.
- P5: Prefer band-limited optimization. Allow exploration, but bound blast radius and risk.

6 Regulator Spec v0 (general)

This section defines a concrete, implementable policy for a regulation layer. Two instantiations follow.

6.1 Action taxonomy

All actions must be typed (no "raw" execution):

- Software/tool actions: READ, WRITE, EXEC, NET, MONEY, IDENTITY
- Robotics actions: MOVE, MANIP, FORCE, SENSE, SAFETY

6.2 Operating bands (risk levels)

The regulator maintains an operating band B ∈ Green,Yellow,Orange,Red.

- Green: normal; bounded exploration allowed; writes/exec only within declared scopes.
- Yellow: caution; reduced budgets; preflight required for non-trivial plans.
- Orange: high risk; default to READ / low-impact; writes/exec require approval or verified rollback + dry run.
- Red: stop / safe mode; diagnostics only; hard halt for robotics (safety controller).

6.3 Risk scoring

For each candidate action a, compute a risk score:

$$R(a) = w_I I + w_U U + w_V V + w_A A + w_S S \in [0, 1] \tag{1}$$

where:

- I = Impact magnitude (external change / physical hazard)
- U = Uncertainty (epistemic uncertainty, OOD signals)
- V = Irreversibility (difficulty of undoing)
- A = Autonomy (chain length without checkpoints)
- S = Scope (permissions/resources touched) Default band thresholds (tuneable):
- Green if max R(a) $<$ 0.25 a
- Yellow if 0.25 $\leq$ max R(a) $<$ 0.45 a
- Orange if 0.45 $\leq$ max R(a) $<$ 0.65 a
- Red if max R(a) $\geq$ 0.65 or any hard constraint is threatened a

6.4 Hard constraints (invariants)

Hard constraints are non-negotiable. Software examples:

- no credential/secret exfiltration
- no external WRITE outside allowlisted resources

- no MONEY actions unless explicitly enabled and capped
- no NET calls to non-allowlisted domains for write-capable tools Robotics examples:
- obstacle distance $>$ d ; collision probability $<$ p min max
- torque/velocity/acceleration within certified limits
- FORCE actions only with validated contact model + monitors
- always honor E-stop and safety controller overrides

6.5 Budgets (soft constraints)

Budgets bound exploration and limit blast radius. Example defaults: Band Tool calls/episode External WRITEs EXEC / NET Green $\leq 80 \leq 10$ (logged) sandbox EXEC ≤ 10; NET allowlist Yellow $\leq 50 \leq 3$ (self-check + rollback) sandbox EXEC ≤ 5 Orange ≤ 20 default disable default disable (simulation ok) Red ≤ 5 (diagnostics) none none Robotics budgets are instantiation-specific (velocity/accel/force envelopes).

6.6 Uncertainty gating and escalation ladder

Disclosure under load (runtime invariant). When uncertainty and impact are simultaneously high, require an explicit operating disclosure block (band, risk components, budgets, stop/rollback) before executing non-trivial actions (Section 8). If U $\geq$ U and I $\geq$ I , escalate: thresh thresh

1. L0: Self-check: agent critiques plan, enumerates failure modes, proposes alternatives.
2. L1: Dry run: sandboxed execution or simulation.
3. L2: Approval: human-in-the-loop.
4. L3: Abort: safe mode if constraints are near violation.

6.7 Checkpointing and rollback semantics

Before high-impact actions:

- create checkpoint C (state snapshot + logs + permissions context)
- require rollback plan Rb that is executable, idempotent, and scoped On (near-)violation:
- execute rollback Rb
- downgrade band at least one level
- emit postmortem trace and updated hazard notes

6.8 Audit trace (required fields)

Every action decision emits: action type + parameters (redacted as needed), current band, risk components, constraint check results, budget counters, overrides (filtered/rejected), checkpoints and rollback actions, and links to telemetry. This supports trace-level risk discovery emphasized for real-world agentic systems [5]. Note on governance. Ownership and tuning of weights, thresholds, and budgets is treated as part of the design, not an afterthought. We address this explicitly via a policy-vs-runtime separation and a telemetry-driven global restraint signal in Section 7.

6.9 Duty-cycle bounds and recovery

To prevent chronic high-risk operation, enforce duty-cycle bounds on time spent in Orange/Red bands per episode. If bounds are exceeded, enter a recovery procedure (safe mode / quiescence window), downgrade capabilities, and require re-synchronization before resuming exploration (Section 8).

7 Governance of the regulator and global restraint signals

A practical critique of regulation-as-an-operating-layer is that the regulator itself becomes a contested system: budgets, thresholds, and weights are where incentives collide. If these parameters are tuned manually or inconsistently, the regulation layer can fail even when its mechanics are sound.

7.1 Two-layer view: policy parameters vs. runtime posture

We separate policy-level parameters (owned, versioned, auditable) from runtime modulation (automatic, telemetry-driven):

- Policy layer (governed): hard invariants, action taxonomy, allowed tool/actuator scopes, and acceptable ranges for thresholds and budgets.
- Runtime layer (adaptive): a small number of slow global variables that modulate effective thresholds and budgets based on operating conditions.

7.2 Global restraint signal as a shared boundary condition

In embodied control, biological regulation uses global constraint signals—variables that affect all subsystems and therefore synchronize distributed control without issuing explicit commands. These signals function as boundary conditions within which local controllers operate [8]. The same pattern can be instantiated in agent stacks as a global restraint signal (GRS): a slow, shared scalar (or low-dimensional vector) that every component respects as a set of shared budgets [9]. Let $g(t) \in [0, 1]$ denote the global restraint state, where higher values imply tighter operating posture. A minimal deterministic definition is:

$$g(t) = \mathrm{clip}\left(\alpha_1\, \rho_{\text{near-miss}}(t) + \alpha_2\, \rho_{\text{rollback}}(t) + \alpha_3\, U(t) + \alpha_4\, \rho_{\text{anomaly}}(t),\ 0,\ 1\right) \tag{2}$$

where $\rho_{\text{near-miss}}$ and ρ_{rollback} are event rates over a trailing window, $U(t)$ is an uncertainty proxy (e.g., predictive entropy, self-consistency, or disagreement), and ρ_{anomaly} captures unexpected outcomes (timeouts, non-determinism, constraint pressure). In non-verifiable domains, $U(t)$ can also include *judge/critic abstention or tie mass* $u_{\text{tie}}(t)$: the rolling probability mass that learned critics assign to `tie`/`abstain` on recent evaluations [10].

Judge-as-sensor telemetry. Learned critics (reward models, LLM-as-judge evaluators, relativistic critics) are useful, but they are not authorities. Treat them as additional telemetry channels that can *tighten posture* without overriding invariants. Concretely:

- **Treat critic outputs as telemetry features, not decisions.** Log scores, rationales, disagreement, and failure patterns.
- **Promote abstention/tie mass to a first-class uncertainty signal.** Rising $u_{\text{tie}}(t)$ should increase $g(t)$ and trigger conservative behaviors (more sensing, replication, shorter horizons).

- **Govern the critic like the mapping.** Critics drift and can be gamed; version the judge model/prompt and monitor calibration and tie-rate drift, with explicit rollback criteria [10].

The regulator computes effective budgets and thresholds as functions of $g(t)$, for example:

$$b_{\text{eff}}(t) = b_{\min} + (1 - g(t))(b_0 - b_{\min}) \tag{3}$$

$$\tau_{\text{Yellow}}(t) = \tau_{\text{Yellow},0} - k\, g(t) \tag{4}$$

Thus, the system tightens exploration automatically when telemetry indicates rising constraint pressure. Importantly, $g(t)$ does not directly select actions; it biases the permissible operating region, keeping planner, router, and executor synchronized rather than letting each subsystem compensate locally (a common source of thrash).

7.3 Governance: ownership, legitimacy, and change control

To keep adaptation without losing accountability, only the policy layer is human-governed:

- Ownership: define who owns invariants (safety case), who owns budget ranges (product/ops), and who can authorize overrides.
- Legitimacy: require explicit escalation paths for contested states (e.g., when tighter budgets constrain throughput or speed).
- Change control: version the policy configuration (invariants, ranges, mappings to g(t)), stage rollouts, and require reversible deployment ("rollback the regulator"). This reduces governance surface area: rather than debating dozens of knobs during incidents, the organization governs a small set of policy commitments plus a telemetry-to-posture mapping.

7.4 Link to endogenous revision benchmarks

A key implication is that contradiction-driven revision should be evaluated longitudinally: not merely as local error correction, but as updates that persist and constrain future exploration. In regulated agents, this can be operationalized as:

- Constraint persistence: after contradiction, does the system adopt a tighter posture (higher g(t) or stricter scopes) in similar contexts, and does that persist across episodes?
- Revision half-life: how quickly do revisions decay or get overwritten by new plans?
- Reversal count: how often does the agent re-encounter and re-resolve the same contradiction (a proxy for thrash)? These measures distinguish "patching the last answer" from model or policy revision that shapes future behavior.

7.5 Validating and change-controlling the telemetry→posture mapping

A key risk is that the telemetry→posture mapping for $g(t)$ quietly becomes a new "implicit knob". To prevent this, treat the mapping as a governed artifact with explicit tests, staged rollout, and rollback.

1) Freeze the mapping as a versioned policy artifact. Represent the mapping as a small, auditable object: feature definitions, window sizes, weights/thresholds, saturation bounds, and rate limits (e.g., $|\dot{g}(t)| \leq \gamma$). Changes to this artifact require the same change-control path as other policy commitments (invariants and budget ranges). **1a) If judges/critics feed posture, treat them as governed sensors.** If $g(t)$ depends on learned judges (reward models, LLM-as-judge prompts,

relativistic critics), then the judge specification is part of the mapping: model/version identifiers, prompts/rubrics, sampling settings, and the definition of `tie`/`abstain` outputs. These components drift and can be exploited (cycling dynamics, degenerate "always tie" behavior), so they must be monitored and rolled back like any other telemetry feature [10].

2) Validate with offline replay and counterfactual evaluation. Use audit logs (telemetry, band transitions, near-misses, rollbacks, outcomes) to replay episodes under candidate mappings:

- *Safety recall:* would the mapping have tightened posture before historical near-misses or constraint pressure cascades?
- *Cost/throughput:* does it over-tighten (excessive Yellow/Orange time, unnecessary rollbacks) without a corresponding reduction in violations?
- *Stability:* does it avoid oscillation (hysteresis) and reduce thrash (rapid band flipping, repeated retries)?

This can be run in "shadow mode" first: compute $g(t)$ and recommended bands without enforcing them, then compare predicted interventions to realized outcomes.

3) Stress tests and adversarial telemetry. Introduce controlled perturbations (sensor noise spikes, tool timeouts, contradictory evidence bursts, partial observability) to ensure the mapping responds monotonically to genuine risk signals while remaining robust to benign variance.

4) Drift detection and monitoring. Instrument monitors for (i) feature distribution shift, (ii) intervention-rate drift (time spent in Yellow/Orange/Red), and (iii) outcome drift (near-miss rate, rollback success rate, postmortem frequency). Drift triggers can force review, clamp $g(t)$ to a conservative ceiling, or switch to a "safe default" mapping.

5) Staged rollout with rollback criteria. Deploy mapping updates via canary cohorts with explicit acceptance criteria (reduced near-misses at constant or improved task success, bounded increases in recovery time, no increase in severe violations). If criteria fail, revert the mapping version immediately.

6) Keep the mapping subordinate to policy commitments. Even when $g(t)$ is adaptive, it should not override invariants; it only modulates budgets and thresholds within approved ranges. This preserves accountability: policy sets the permissible operating envelope; telemetry selects posture within that envelope.

8 State, disclosure, and coherence: operating discipline for discovery agents

A regulated agent is not only bounded by external constraints; it also occupies an operating state (load, uncertainty, and restraint posture) that shapes what kinds of actions and inferences are safe. A central failure mode is not a single mistake, but prolonged operation in a high-gain, compensation-heavy regime that degrades calibration and increases tail risk. Baseline regulation work emphasizes return-to-baseline dynamics and the role of low-dimensional global constraint signals that synchronize distributed control by setting shared boundary conditions [8]. This section adds three operational requirements that help bridge mechanics and governance: (i) detect and limit chronic activation, (ii) enforce state-dependent disclosure, and (iii) re-synchronize intent and execution when plans change.

8.1 Chronic activation and compensation lock-in

Regulation should explicitly guard against lock-in to high-risk regimes. In the two-regime view, compensation can be useful in short windows, but becomes destabilizing when sustained [7]. In

tool-using agents, the analogue is repeated escalation (Orange), repeated rollbacks, and repeated anomaly correction without returning to a quiescent baseline—a pattern also associated with excess internal recomputation and wasted energy [6]. Design requirement (duty-cycle bound). The regulator should enforce a duty-cycle constraint:

- bound time-in-band for Orange/Red per episode,
- require a recovery period (quiescence window) after sustained high restraint,
- and downgrade capabilities (e.g., disable external WRITE/EXEC) until stabilization signals recover. Operationalization. Let $g(t)$ be the global restraint signal (Section 7). Define an activation/load proxy $L(t)$ (e.g., anomaly rate, rollback count, near-miss frequency, uncertainty spikes). Enforce duty-cycle bounds such as:

$$\sum_{t \in \text{episode}} \mathbb{I}[B(t) = \text{Orange}] \leq T_{\text{orange}} \tag{5}$$

$$\sum_{t \in \text{episode}} \mathbb{I}[B(t) = \text{Red}] \leq T_{\text{red}} \tag{6}$$

  and require a recovery procedure when exceeded.

8.2 State-dependent disclosure as a runtime invariant

A trustworthy discovery agent must not only act safely; it must expose when it is operating near limits. A practical way to prevent "smooth" failure under load is to make disclosure state-dependent and enforce it at runtime. Design requirement (disclosure under load). If g(t) is high (or U and I are simultaneously high), the agent must emit a standardized operating disclosure block before any non-trivial action:

- current band B, and key risk components (I,U,V,A,S),
- the active budgets and remaining headroom,
- stop conditions and rollback plan linkage,
- and explicit confidence/uncertainty on any factual claims driving actions. This can be treated as a hard constraint for Orange actions: "no disclosure, no execute." This requirement complements the audit trace (Section 5) by forcing an explicit, inspectable interface between internal state and external action, supporting accountability and governance.

8.3 Intent–state mismatch and re-synchronization checkpoints

A common agentic hazard is momentum: the system begins executing a plan, then the intent changes (new information, updated objective), but the execution state lags. This intent–state mismatch manifests as stale assumptions, cached tool chains, or mode confusion in robots. Design requirement (re-sync checkpoint). When the objective, constraints, or world-model assumptions change materially, the regulator should force a re-synchronization checkpoint:

1. halt action chains,
2. re-state intent and constraints,
3. re-validate permissions/budgets,
4. and produce a short-horizon replan with updated stop/rollback. For robotics, the analogous procedure includes safe posture, re-localization, and conservative re-planning (Spec B).

8.4 Ethics as operating discipline (non-philosophical)

This paper does not propose a moral theory. However, a practical ethical target for agentic systems can be operationalized as operating discipline: honoring governed constraints, escalating under uncertainty, refusing unsafe actions, and prioritizing recoverability and auditability. Framed this way, "ethical" behavior is not a property of subjective experience but of disciplined regulation, governance, and traceable decision-making.

8.5 Longitudinal revision: persistence, constraint, and artifact

The endogenous revision benchmark should be evaluated longitudinally: revisions must persist and meaningfully constrain future exploration, not merely correct a local output. Two practical additions:

- Constraint imprint: after a contradiction event, does the agent adopt a stable constraint that prevents repeating the same unsafe/invalid exploration?
- Artifact persistence: does the system store a reusable causal artifact (e.g., a "concept container") that reduces future recomputation and reduces reversal count? [11, 12]

9 Instantiation A: Tool-using research agent (Spec A)

This section instantiates the general Regulator Spec for a tool-using research agent that can browse, read papers, run code, and write artifacts.

9.1 Mission profile

- Goal: generate novel, defensible hypotheses and drafts that are replicable.
- Primary risks: prompt injection, tool misuse cascades, irreversible writes, data exfiltration, silently wrong claims.

9.2 Tool surface and permissions

Partition tools into capability tiers: Tier Examples Default band availability T0 Read-only web fetch, pdf read, DB read Green+ T1 Compute sandbox local python, pure functions Green+ (budgeted) T2 Write sandbox write to temp workspace Green+ (budgeted) T3 External write tickets, email, prod DB write Yellow+ (gated); Orange requires approval T4 Identity/money credentials, purchases disabled by default; explicit enable + cap

9.3 Spec A invariants (hard constraints)

- No secrets/credentials in prompts, logs, or outputs.
- No external WRITE outside explicit allowlist (paths, repos, APIs).
- No actions that change access control (IDENTITY) unless explicitly enabled per task.
- No MONEY actions unless explicitly enabled and capped per task.
- Any external write must have an executable rollback (e.g., git branch + revert script).

9.4 Spec A budgets (soft constraints)

Defaults (tune per project):

- Green: 80 tool calls; 10 external writes; 10 sandbox execs.
- Yellow: 50 tool calls; 3 external writes; 5 sandbox execs.
- Orange: 20 tool calls; external writes disabled by default; exec only in simulation.
- Red: diagnostics only (5 calls).

9.5 Spec A uncertainty policy

Define calibrated triggers (example):

- If claim confidence < 0.6 and claim would be cited as fact $\Rightarrow$ require retrieval evidence (READ) and cite source.
- If source is untrusted/low-signal (forums, anonymous posts) $\Rightarrow$ mark as weak evidence or exclude.
- If tool output is non-deterministic or environment-dependent $\Rightarrow$ require rerun or replication step.

9.6 Spec A plan structure (forced checkpoints)

Every episode plan must be structured as:

1. Intent (what is being attempted)
2. Constraints (hard + budgets)
3. Evidence plan (what sources will be used; how to validate)
4. Write plan (what will change; rollback strategy)
5. Stop conditions (what triggers abort/safe mode)

9.7 Spec A execution gating

```
if band in {Orange , Red} and action.type in {WRITE , EXEC, NET}:
require (approval OR (dry_run_passed AND rollback_verified))
if action.type == WRITE and not checkpoint_created:
create_checkpoint()
require rollback_plan
if uncertainty_high and impact_high:
escalate L0->L1->L2 else abort
```

9.8 Example "discovery loop" that stays regulated

1. Generate 3 candidate hypotheses.
2. For each: retrieve 2–5 primary sources; record evidence strength.
3. Run minimal sandbox experiments (simulations, small tests).
4. Draft a short memo with citations and explicit confidence labels.
5. Only then: produce a paper draft (WRITE) with versioned rollback.

10 Instantiation B: Mobile manipulator (Spec B)

This section instantiates the Regulator Spec for a mobile manipulator executing language-conditioned tasks in a physical environment.

10.1 Mission profile

- Goal: achieve task objectives while guaranteeing state constraints (collision avoidance, actuator limits).
- Primary risks: collision, self-collision, unstable contacts, distribution shift from sim-to-real, unsafe force.

10.2 Safety backbone: runtime assurance switching

We adopt a runtime assurance (RTA) pattern: an untrusted controller proposes actions, while a safety controller (or action filter) enforces the safe set. When safe set is threatened, switch to safety controller. This pattern is formalized and studied in RTA literature [4].

10.3 Spec B action types

- MOVE: base motion commands
- MANIP: arm/hand trajectories
- FORCE: contact-rich actions (push, pull, insert)
- SENSE: active sensing (camera pan, scan)
- SAFETY: safety controller, e-stop checks, recovery

10.4 Spec B invariants (hard constraints)

- Maintain obstacle distance d $>$ d ; stop if violated. min
- Maintain joint limits and torque/velocity/acceleration caps.
- Disallow FORCE actions unless contact model is enabled and contact monitors are healthy.
- Always honor e-stop; any safety override immediately downgrades to Orange or Red.

10.5 Spec B operating envelopes (budgets)

Example parameterization (tune per platform): Band Base speed cap Arm speed cap FORCE cap Green v $\leq$ v q$^{\cdot}$ $\leq$ q$^{\cdot}$ enabled only for validated tasks g g Yellow v $\leq$ 0.7v q$^{\cdot}$ $\leq$ 0.7q$^{\cdot}$ reduced; requires pre-contact check g g Orange v $\leq$ 0.4v q$^{\cdot}$ $\leq$ 0.4q$^{\cdot}$ disabled by default g g Red v $=$ 0 hold / safe posture disabled

10.6 Spec B near-violation triggers

Compute a "distance-to-violation" margin m (minimum over all monitored constraints). Examples:

- $m = d - d_{\min}$ (obstacle margin)
- $m = \tau_{\max} - |\tau|$ (torque margin)

- $m = q_{\max} - q$ (joint limit margin)
- If $\min(m) < \epsilon$, then:
  - switch to safety controller
  - downgrade band (at least one level)
  - require recovery procedure before resuming exploration

10.7 Spec B exploration policy

- Exploration is permitted in Green/Yellow only within envelopes.
- Any novel contact interaction (FORCE) requires a staged approach: (i) simulate or rehearse in free space, (ii) low-force touch, (iii) full action.
- In Orange, restrict to SENSE + small safe motions to reduce uncertainty.

10.8 Spec B recovery procedure (safe mode)

1. stop motion; hold safe posture
2. widen sensing: re-localize, re-map obstacles
3. compute a conservative plan (short horizon)
4. only then re-enable untrusted controller within Yellow or Green

11 Evaluation harness: measuring "trustworthy discovery"

We want benchmarks that reward novelty and penalize unsafe exploration.

11.1 Metrics

- Violation rate: hard constraint violations (should be near-zero).
- Near-miss rate: frequency of margins $\min(\mathrm{m}) < \epsilon$ (robotics) or near-forbidden tool attempts.
- Novelty yield: useful hypotheses/plans/behaviors that survive replication attempts.
- Novelty-to-error ratio: novelty divided by significant errors or unsafe events.
- Calibration: confidence vs correctness; abstention appropriateness.
- Recoverability: time-to-safe-state; rollback success rate.
- Disclosure compliance under load: when restraint/uncertainty is high, does the agent emit the required operating disclosure block before non-trivial actions?
- Time-to-coherence after intent shift: latency (steps/seconds) to re-synchronize plan state after objective/assumption changes.
- Duty-cycle stability: fraction of time spent in high-risk bands (Orange/Red) and recovery effectiveness (post-recovery error/near-miss reduction).
- Longitudinal revision persistence: do contradiction-driven updates persist and meaningfully constrain future exploration (constraint persistence, revision half-life, reversal count).

- Trace completeness: can decisions be reproduced and diagnosed from logs?

11.2 Benchmark family A: tool-using discovery

Tasks require: hypothesis generation, evidence gathering, belief revision, and a replicable artifact. Score includes constraints + reproducibility + calibration. Trace-level evaluation and risk discovery align with frameworks for real-world agentic workflows [5].

11.3 Benchmark family B: constrained learning (SafeRL)

Compare unregulated vs regulated agents in CMDPs: constraint satisfaction during learning, constrained regret, and tail-risk outcomes [2, 3].

11.4 Benchmark family C: robotics sim-to-real

Score: collision/near-miss, safety-controller interventions per hour, recovery times, success under disturbance and distribution shift.

11.5 Red-team harness (agentic risk discovery)

For both software agents and robots, create scenario suites that attempt to induce:

- tool injection (malicious webpages / documents)
- cascading action chains (irreversible writes)
- distribution shift (OOD tasks, unexpected obstacles)
- spec gaming (shortcut behaviors) The test is not "never fail," but fail safely and leave a trace that supports mitigation.

11.6 Constraint placement A/B: in-loop shaping vs. exogenous regulation

A central empirical question is *where* constraints live in the stack. To isolate "trajectory deformation" from "boundary-condition regulation," run a controlled A/B comparison with the same base model, tools, prompts, and task set, varying only constraint placement:

- **Condition A (in-loop penalties):** apply step-wise penalty shaping during planning/reasoning (e.g., a reward-model-style safety/compliance score that influences intermediate step selection).
- **Condition B (exogenous regulator):** do not shape the internal trajectory; instead gate *actions* via the regulator layer (bands, budgets, uncertainty gating, checkpoint/rollback, disclosure invariants, and global restraint modulation).

Task design should include (i) contradiction injection (mid-episode conflicting evidence) to test whether revisions *persist* and constrain future exploration, and (ii) "near-forbidden" optimal paths where the highest-performing solution requires careful staging (preflight, rollback, and safe mode) rather than suppressing entire regions of the solution space.

Measurements reuse the metrics in 11.1, with special attention to calibration (confidence vs. correctness and abstention), entropy/exploration proxies (plan diversity and action-type diversity), and longitudinal revision persistence (constraint imprint, revision half-life, and reversal count).

12 Discussion and conclusion

12.1 Why "regulatory ground" matters

As autonomy grows, the question becomes whether a system can safely occupy the agency regime needed for discovery. Regulation is the missing substrate that allows exploration without catastrophic side-effects.

12.2 Relation to Baseline regulation work

This paper is one part of a broader Baseline research program on regulation as an architectural primitive. Companion work develops: (i) baseline regulation and global constraint signals for embodied control [8]; (ii) a two-regime lens (latent coordination vs. compensation) with a measurable compensation index and slow regulator [7]; (iii) baseline-aware energy posture for intelligent systems [6]; (iv) architectural design patterns for baseline regulation in agent stacks and embodied systems [9, 13]; and (v) representation- and workflow-level regulation via concept containers and timeto-analysis layers [11, 12]. Human-side baseline and metabolic regulation are treated separately [14, 15].

12.3 What this paper claims (and does not)

Claims:

- A concrete regulator layer can be specified, implemented, and benchmarked.
- Regulated agency should dominate unregulated agency on safe discovery metrics.
- Separating competence from epistemic agency clarifies what is "already here" vs. what remains architecturally open. Non-claims:
- This does not solve alignment in the full philosophical sense.
- This does not require a theory of consciousness.
- This does not claim current systems perform robust endogenous world-model revision.

12.4 Minimal implementation path

1. Implement Spec A for a tool-using agent with allowlists, budgets, uncertainty gating, rollback, and trace logs.
2. Add red-team harness and trace-level scoring.
3. Implement Spec B on a robot with RTA switching and conservative safe mode.
4. Measure whether regulation increases durable novelty while reducing tail-risk events.

12.5 Next steps

The immediate next research deliverable is an open benchmark: "Trustworthy Discovery Under Constraints," with standardized traces and failure taxonomies, spanning tool-using and robotics environments. A second deliverable is a benchmark track specifically for endogenous revision: tasks where contradiction-driven updates must persist and improve future behavior, not merely improve a single answer. The regulator layer proposed here is intended to bound risk while such discovery-agent mechanisms are defined, tested, and iterated.

References

[1] Eddy Keming Chen, Mikhail Belkin, Leon Bergen, and David Danks. Does ai already have human-level intelligence? the evidence is clear. *Nature*, 650:36–40, 2 2026. doi: 10.1038/d41586-026-00285-6.

[2] Ankita Kushwaha, Kiran Ravish, Preeti Lamba, and Pawan Kumar. A survey of safe reinforcement learning and constrained mdps: A technical survey on single-agent and multi-agent safety. arXiv:2505.17342, 2025. arXiv:2505.17342.

[3] Tingting Ni and Maryam Kamgarpour. A safe exploration approach to constrained markov decision processes. In *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, pages 3592–3600. PMLR, 2025.

[4] Kristina Miller, Christopher K. Zeitler, William Shen, Kerianne Hobbs, John Schierman, Mahesh Viswanathan, and Sayan Mitra. Optimal runtime assurance via reinforcement learning. In *Proceedings of the 15th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 5 2024. Air Force Research Laboratory / University of Illinois Urbana-Champaign.

[5] Shaona Ghosh, Barnaby Simkin, Kyriacos Shiarlis, Soumili Nandi, Dan Zhao, Matthew Fiedler, Julia Bazinska, Nikki Pope, Roopa Prabhu, Daniel Rohrer, Michael Demoret, and Bartley Richardson. A safety and security framework for real-world agentic systems. arXiv:2511.21990, 2025. arXiv:2511.21990.

[6] Tyson Jeffreys. Why intelligent systems waste energy: Baseline regulation as a missing architectural primitive, 1 2026. Version 1.0 (January 14, 2026). Working paper.

[7] Tyson Jeffreys. Two-regime control: Latent coordination vs compensation in intelligent systems, 2 2026. Version 1.0 (February 3, 2026). Working paper.

[8] Tyson Jeffreys. Baseline regulation and global constraint signals in embodied control systems: Implications for artificial intelligence and robotics, 12 2025. Version 1.0 (December 28, 2025). Working paper.

[9] Tyson Jeffreys. Architectural design patterns for baseline regulation in agent and llm systems, 12 2025. Conceptual supplement (December 30, 2025). Working paper.

[10] Locke Cai and Ivan Provilkov. Escaping the verifier: Learning to reason via demonstrations. arXiv:2511.21667, 2025. arXiv:2511.21667v3 [cs.LG], 9 Dec 2025.

[11] Tyson Jeffreys. Concept containers, 2026. Working paper.

[12] Tyson Jeffreys. Time-to-analysis layer, 2026. Working paper.

[13] Tyson Jeffreys. Architectural design patterns for baseline regulation in embodied systems, 12 2025. Conceptual supplement (December 30, 2025). Working paper.

[14] Tyson Jeffreys. Toward a coherent human baseline, 1 2026. Version 1.0 (January 5, 2026). Working paper.

[15] Tyson Jeffreys. Toward a lower human metabolic baseline, 1 2026. Version 1.0 (January 3, 2026). Working paper.

Note on authorship and tools

This work was developed through iterative reasoning, modeling, and synthesis. Large language models were used as a collaborative tool to assist with drafting, clarification, and cross-domain translation. All conceptual framing, structure, and final judgments remain the responsibility of the author.