# Phase Discipline for Regulated Agents

## Transition Windows and State–Action Alignment

*Agent architectures / robotics / regulated discovery*

Tyson Jeffreys
Independent Researcher
tyson@staygolden.dev

Version 0.1 — February 12, 2026

**Series note.** This paper extends the Baseline regulation series by naming a missing runtime layer: *phase discipline*. Prior papers specify (i) a regulator layer with bands, budgets, rollback, and a telemetry-driven global restraint signal $g(t)$ for trustworthy discovery [1, 2]; (ii) representation-level commits via concept containers [3]; (iii) a research-systems objective centered on reusable analysis artifacts and bursty synthesis [4]; and (iv) verifier-gap governance where learned critics are treated as sensors, with abstention as uncertainty telemetry and commits gated under non-verifiable selection [5]. Here we add a second runtime control variable alongside posture: a phase state $p(t)$ that schedules work types and reserves transition windows for consolidation and commits.


## Abstract

Band-limited regulation constrains *how risky* an agent may operate (budgets, allowed actions, rollback), but it does not specify *when* different classes of cognitive work should occur. In practice, many failures are not outright constraint violations; they are **state–action mismatches**: deep planning during periods of high internal contention, commitment under unstable selection signals, or exploration when the system is already compensating.

We propose **phase discipline**: a runtime layer that tracks a small phase state $p(t)$ (restore, transition, act) and enforces **state–action alignment** by gating action classes, planning depth, and commit rights based on phase. The key primitive is the **transition window**: a stable interval reserved for consolidation (analysis artifacts), representation commits (concept containers), and policy updates, with **bounded override** as a priced, recoverable mechanism for deliberate phase-breaking.

We specify a minimal phase controller, relate it to Two-Regime Control (latent coordination vs. compensation), show how it composes with the global restraint signal $g(t)$ and verifier-gap selection telemetry, and propose measurable metrics and A/B experiments in tool-using and robotics settings.


---

# 1 1. Motivation: beyond bands

## 1.1 1.1 The missing failure mode

In *Regulatory Ground*, trustworthy discovery is framed as a band-limited optimization problem: as capability and actuation increase, failures shift from isolated errors to cascades, and the agent requires an explicit regulator layer (bands, budgets, uncertainty gating, checkpoint/rollback, and safe mode) [1]. That layer controls *risk posture* via a telemetry-driven global restraint signal $g(t)$.

However, many real failures arise even when $g(t)$ is conservative:

- **Thrash**: repeated recomputation and reversals (plan churn) without violating a hard constraint.
- **Chronic compensation**: the system sustains a high-activation mode longer than necessary, accumulating error and fatigue-like artifacts [2].
- **Premature commits**: durable writes (documents, policies, representations) made while selection is unstable or evidence is weak [3].

These are not primarily "unsafe actions"; they are *operating discipline* failures. The system is doing the wrong kind of work at the wrong time.

### 1.2 1.2 A second control variable

We propose adding a second low-dimensional runtime controller:

- **Posture** $g(t)$: how tightly constrained the system should be (budgets, allowed actions, escalation).
- **Phase** $p(t)$: what operating mode the system should be in (act, restore, transition), which determines what *classes of work* are permitted now.

The core thesis:

> **Trustworthy discovery requires not only band limits on risk but phase discipline that schedules work to minimize internal contention, bound compensation duty cycles, and reserve stable transition windows for commits.**

## 2 2. Definitions

### 2.1 2.1 Phase variable

Let $p(t)$ be a discrete phase state. A minimal taxonomy:

- **Act**: high throughput execution (tool use, exploration, task completion).
- **Restore**: quiescence/recovery (monitoring, low-cost maintenance, minimal tool actions).
- **Transition**: a bounded window for consolidation and commits (analysis artifacts, representation updates, policy writes).

Optional extensions (implementation-dependent):

- **Override**: deliberate phase-breaking with tightened budgets and mandatory recovery.
- **Safe mode**: regulator-enforced fallback (already present in Regulatory Ground).

### 2.2 2.2 Action classes

Phase discipline works by gating *action classes*. Example classes:

- **Explore**: widen hypotheses, sample plans, gather evidence.
- **Exploit/Execute**: commit to a plan, act on the environment/tools.
- **Consolidate**: compress state into analysis artifacts (levers, predictions, falsifiers).
- **Commit**: durable writes: external actions with side effects, policy updates, or representation commits (containers).

### 2.3 2.3 State–action mismatch

Define **state–action mismatch** as a situation where the agent executes an action class that is disallowed or high-cost for its current phase. Mismatch is distinct from constraint violation:

- constraints are about *external safety / budgets*;
- mismatch is about *internal stability / contention / thrash.*

Mismatch is the phase analog of Two-Regime Control's compensation regime: it is the condition under which the system tends to burn compute/energy and accumulate errors [6, 2].

# 3 3. The transition window

## 3.1 3.1 Why transitions matter

The transition window is the key new primitive. It is not merely "rest" and not merely "execution." It is a bounded interval reserved for:

- **Consolidation**: producing analysis artifacts that expose causal structure and falsifiers (Time-to-Analysis) [4].
- **Commit gating**: permitting durable writes only when uncertainty telemetry is favorable.
- **Representation commits**: creating/updating concept containers as stable compressed structure [3].

Intuition: commits are cheapest and safest when internal contention is lowest. In the absence of verifiers, commits must be coupled to uncertainty telemetry (abstention/tie mass, disagreement, stability) rather than confidence narratives [5].

## 3.2 3.2 Phase discipline as "baseline operation" for agents

In the Baseline series, "baseline" means operating near a stable attractor with minimal chronic compensation. Phase discipline operationalizes this for agents by:

- bounding time spent in high-activation action phases,
- enforcing recovery/quiescence,
- and concentrating irreversible commitments into controlled windows.

# 4 4. Composition with runtime regulation

## 4.1 4.1 Orthogonality: $g(t)$ vs $p(t)$

A simple way to compose them:

- $g(t)$ sets *tightness*: budgets, allowed tools, maximum plan depth.
- $p(t)$ sets *work type*: what classes of work are allowed now (execute vs consolidate vs recover).

Example: the system can be in phase **Act** but with high restraint (tight posture)—allowing only short-horizon execution and forbidding irreversible writes. Conversely, the system can be in phase **Transition** with low restraint, enabling deeper consolidation and broader commit rights.

## 4.2 4.2 Telemetry inputs

Phase and posture should be driven by similar telemetry, but interpreted differently.

**Posture** $g(t)$ uses telemetry to answer: *How risky is it to act?*

**Phase** $p(t)$ uses telemetry to answer: *Is the system stable enough to consolidate/commit, or should it restore, or act?*

Candidate telemetry (existing and new):

- **Constraint proximity**: budget exhaustion, near-miss rates, safety-controller interventions (Regulatory Ground).
- **Compensation index**: internal thrash proxies, reversal rate, tool-call churn (Two-Regime / Waste Energy).

- **Selection uncertainty**: abstention/tie mass, disagreement entropy, judge stability (Verifier Gap / judge-as-sensor).
- **Recovery markers**: whether the system returns to low-thrash operation after load.

### 4.3 4.3 Hysteresis and dwell time

Agents can change phase rapidly. That is a feature, but it introduces a known control hazard: **chattering**. Phase transitions should include:

- **hysteresis**: different thresholds for entering vs exiting phases,
- **dwell-time constraints**: minimum time in a phase before switching again.

This mirrors Regulatory Ground's emphasis on stability and avoiding oscillation in posture control.

## 5 5. A minimal phase controller

### 5.1 5.1 Phase scheduler

Define a phase scheduler:

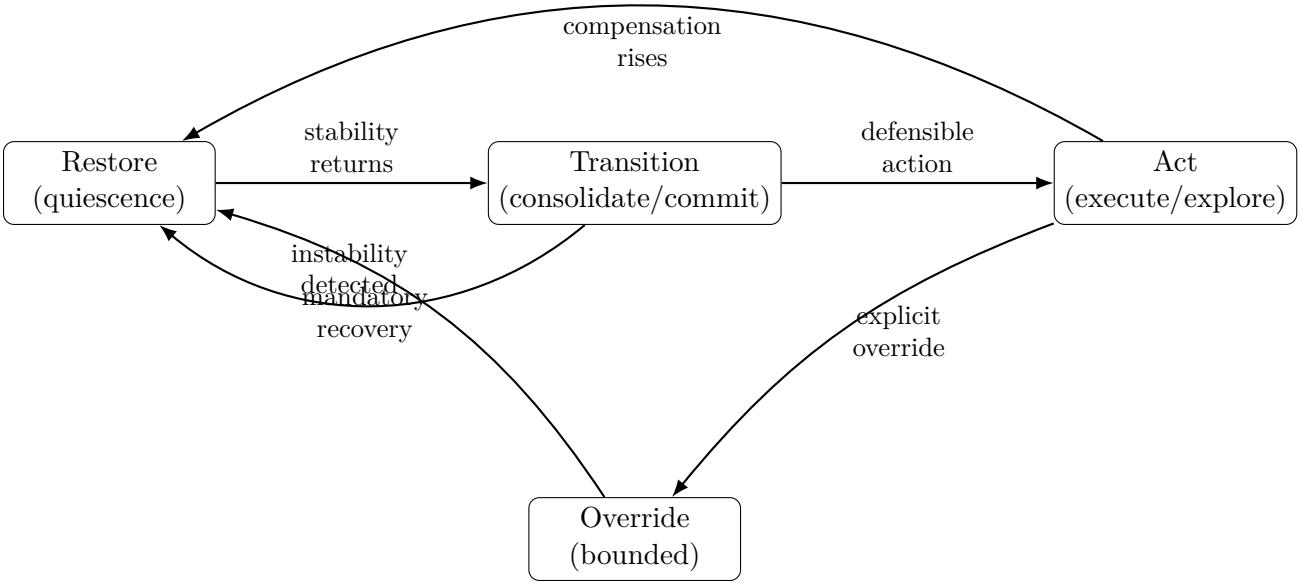
$$p(t) = \pi(\text{telemetry},\ g(t),\ \text{task context})$$

where $\pi$ is a small state machine (or hybrid controller) with conservative transitions.

A minimal policy:

- Enter **Restore** when compensation markers rise (thrash, reversal rate) or when budgets approach limits.
- Enter **Transition** after restore when stability markers return (low thrash, stable selection telemetry).
- Enter **Act** after transition when the next action is defensible (adequate evidence, low uncertainty).

### 5.2 5.2 State machine sketch



### 5.3 5.3 Action gating table

A simple action-permission matrix:

| Action class | Restore | Transition | Act |
|---|---|---|---|
| Monitor / low-cost checks | ✓ | ✓ | ✓ |
| Evidence gathering (bounded) | ∼ | ✓ | ✓ |
| Deep planning / broad search | × | ∼ | ✓ |
| Consolidation (analysis artifacts) | ∼ | ✓ | ∼ |
| Durable commits (writes/containers/policy) | × | ✓† | × |

†Commit is gated by uncertainty telemetry and posture: if abstention/tie mass is high, commits are blocked and evidence is required.

## 6 6. Bounded override

### 6.1 6.1 Why override exists

Discovery often requires short deviations from phase alignment: pushing a difficult plan through, taking an expensive measurement, or exploring a risky branch. Phase discipline should not prohibit this; it should *price and bound* it.

### 6.2 6.2 Override policy

Override is an explicit mode with:

- a declared objective and time budget,
- tightened posture $g(t)$ (reduced tool rights, capped depth),
- and a mandatory recovery path (return to Restore, then Transition).

Override is the macro analog of Two-Regime Control's compensation regime: it is sometimes useful, but it must be duty-cycle bounded and recoverable [2].

### 6.3 6.3 Required disclosure

In verifier-free settings, override should emit a standardized disclosure block before non-trivial actions:

- what is being overridden (phase alignment),
- what uncertainty is present,
- what the bounded budget is,
- and what recovery/rollback is available.

This mirrors the operating disclosure invariant introduced in Regulatory Ground and strengthened by judge-as-sensor telemetry.

# 7 7. Commits as regulated events

## 7.1 7.1 Commit types

A **commit** is any irreversible or persistent change:

- external writes with side effects,
- policy/config updates that alter future behavior,
- representation writes (concept container create/update),
- long-lived analysis artifacts used as downstream inputs.

## 7.2 7.2 Commit gating in the transition window

Phase discipline proposes a simple rule:

> **Commits are only permitted in Transition, and only when uncertainty telemetry is favorable.**

This ties together:

- **Concept Containers**: container writes are commits [3].
- **Time-to-Analysis**: analysis artifacts are reusable structure; they should be produced when stable [4].
- **Verifier Gap**: if tie/abstain mass is high, do not commit—seek discriminating evidence [5].

# 8 8. Implementation sketches

## 8.1 8.1 Tool-using research agent

A regulated research agent can be structured as:

- Act: bounded evidence gathering and tool calls (search, fetch, parse).
- Restore: low-cost monitoring and inventory of uncertainty.
- Transition: produce analysis artifacts, run bounded selection (tournament), and commit a container or recommendation if permitted.

## 8.2 8.2 Robotics: action vs consolidation

In robotics, phase discipline maps naturally onto multi-rate stacks:

- high-rate control continues continuously,
- low-rate planners and policy updates are phase gated,
- commits (map updates, policy changes) are reserved for stable windows.

The transition window is where to perform:

- map/scene consolidation,
- belief revision after contradictions,
- safe policy updates,
- and checkpointing.

# 9 8. Implementation sketches

## 9.1 8.1 Tool-using research agent

A regulated research agent can be structured as:

- Act: bounded evidence gathering and tool calls (search, fetch, parse).
- Restore: low-cost monitoring and inventory of uncertainty.
- Transition: produce analysis artifacts, run bounded selection (tournament), and commit a container or recommendation if permitted.

## 9.2 8.2 Robotics: action vs consolidation

In robotics, phase discipline maps naturally onto multi-rate stacks:

- high-rate control continues continuously,
- low-rate planners and policy updates are phase gated,
- commits (map updates, policy changes) are reserved for stable windows.

The transition window is where to perform:

- map/scene consolidation,
- belief revision after contradictions,
- safe policy updates,
- and checkpointing.

# 10 9. Metrics and experiments

## 10.1 9.1 Metrics

Phase discipline enables new measurable quantities:

- **Phase mismatch rate**: fraction of actions executed in disallowed/high-cost phases.
- **Override duty cycle**: time in override divided by wall time.
- **Recovery half-life**: time to return to low-thrash operation after override.
- **Commit timing quality**: commits per stable-transition window vs commits in act/restore.
- **Thrash reduction**: reversal count, tool-call churn, plan variance (Waste Energy / Two-Regime proxies).
- **Novelty yield under regulation**: durable novelty per unit of override budget (Regulatory Ground evaluation style).

## 10.2 9.2 A/B tests

Two simple experiments:

1. **Phase discipline on/off**: same agent, same tools, same tasks; only phase gating differs. Measure thrash, commit error rate, and recovery.

2. **Commit-in-transition vs commit-anytime**: allow container writes and policy updates only in Transition vs anywhere. Measure contamination (revision half-life, reversal count) and downstream performance.

## 11 10. Discussion: "harmonic" operation is not benevolence

Phase discipline pushes systems toward lower oscillation, lower thrash, and bounded compensation duty cycles. This can look like "harmonic" behavior: stable trajectories with rapid recovery after perturbation.

But **harmonic operation is not a moral guarantee**. Benevolence still depends on:

- the invariants and objectives the regulator enforces,
- commit governance (what the system is allowed to change),
- and traceability/auditability.

The correct claim is narrower and testable: *phase discipline reduces internal contention and makes discovery more sustainable and recoverable.*

## 12 11. Conclusion and next steps

We introduced phase discipline as a missing runtime layer in regulated agents: a discrete phase variable $p(t)$ that schedules work types and reserves transition windows for consolidation and commits. It composes naturally with the global restraint signal $g(t)$ (risk posture) and with verifier-gap governance (judge-as-sensor uncertainty telemetry).

Immediate next steps:

1. Add a phase mismatch failure mode and phase metrics to the Regulatory Ground benchmark harness.
2. Patch Concept Containers to make abstention-gated container writes explicit (commit as regulated event).
3. Implement a minimal phase scheduler in a tool-using agent and measure thrash reduction and commit quality.
4. Extend to robotics stacks by phase-gating low-rate planning and durable updates.

## References


[1] 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.

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

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

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

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

[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.

## 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.