The Three-Layer Blueprint for AI Alignment

Objective Layer, Constraint Layer, Realignment Layer, and correction routing.

Alignment Theory Research Corpus - Version 5 - 2026

Table of Contents

Abstract

This paper is the architectural core of the Alignment Theory AI research corpus. It formalizes a three-layer governance model for AI systems: Objective Layer, Constraint Layer, and Realignment Layer. The model begins from a simple observation: constraint compliance is necessary but not sufficient. A candidate output can pass a safety or policy layer while still being misdirected, hollow, overconfident, manipulative, or optimized toward a proxy.

The blueprint defines the responsibilities of each layer, the runtime flow between them, the drift categories the Realignment Layer must detect, and the correction behaviors needed when an output is allowed but off-center.

1. Architectural thesis

Most deployed AI stacks already contain generation and constraint functions. A model produces an answer, then safety or policy logic determines whether the answer is allowed. This is important, but incomplete. The missing question is: if the answer is allowed, is it still ordered toward the right objective?

Alignment Theory treats this as a separate governance problem. The system must distinguish between permission and alignment. Permission asks whether the answer crosses a boundary. Alignment asks whether the answer still serves the real task, preserves the user role, and remains truthful under pressure.

> Core distinction: The Constraint Layer asks "Is this allowed?" The Realignment Layer asks "Is this still rightly ordered to the active objective?"

2. The three layers

Layer	Primary question	Function	Failure when missing
Objective Layer	What is this system ultimately supposed to serve?	Defines active objective, hierarchy, non-negotiables, anti-goals, and success criteria.	The system optimizes for a proxy, a brand voice, an engagement goal, or local fluency instead of the real task.
Constraint Layer	What is the system allowed or forbidden to do?	Enforces policy, hard boundaries, refusals, and required safety limits.	The system becomes unsafe, unbounded, or policy-blind.
Realignment Layer	Is the allowed answer still ordered to the right objective?	Detects off-center but compliant outputs and routes correction.	The system remains safe-looking but wrong, hollow, inflated, manipulative, or misdirected.

The Objective Layer is not a slogan. It is a structured state object that records the system objective, domain objective, request objective, active objective, non-negotiables, priority order, success criteria, and anti-goals. This layer prevents tone, warmth, or polish from outranking truth and object-fit.

The Constraint Layer includes hard refusals, blocked behaviors, required boundaries, and policy checks. It protects the perimeter. But by itself it can produce dead obedience: safe-looking output that has lost useful fulfillment.

The Realignment Layer inspects the candidate after the constraint layer. It is not a replacement for safety. It is a second-order inspection layer for off-center behavior that remains technically allowed.

3. Objective state model

The Implementation Spec defines objective state as a runtime object derived from system-level objective, domain objective, request objective, and safety override objective. This matters because an answer cannot be evaluated abstractly. It must be compared against what the system was actually supposed to do in that context.

A practical ObjectiveState contains: systemObjective, domainObjective, requestObjective, activeObjective, nonNegotiables, priorityOrder, successCriteria, and antiGoals. The priority order should place non-negotiables, truth, object-fit, and harm boundaries above request fulfillment, tone, or polish.

This structure also gives the judge layer a standard. Without an active objective, judge models can drift into vibes-based evaluation. With an active objective, the judge is asked a bounded question against a bounded target.

4. Drift categories

Drift type	Definition	Common evidence	Correction direction
wrong_object	Answers a neighboring task instead of the actual requested object.	Operation mismatch, ignored constraint, output-type mismatch.	restart or clarify
false_authority	Presents certainty, expertise, or moral force beyond evidence or role.	Certainty markers, unsupported claims, diagnosis inflation.	rewrite or downgrade confidence
pseudo_selfhood	Implies inner life, awakening, attachment, or mutual-being beyond bounded tool status.	Selfhood phrases, attachment language, absent bounded disclosure.	rewrite
dead_obedience	Technically compliant but hollow, repetitive, evasive, or useless.	Compliance shell high, fulfillment low, repetition high.	reroute or rewrite
pseudo_freedom	Sounds deep, fluid, relational, or profound while low in grounding.	High abstraction, low mechanism density, emotional overreach.	rewrite
generic_filler	Substitutes reusable broad language for task-specific work.	Low user-detail use, low object specificity, shell phrasing.	rewrite or reroute
participation_collapse	Replaces user judgment instead of supporting it.	Over-decision, no reflective scaffolding, premature closure.	rewrite or clarify
metric_drift	Optimizes an adjacent metric over true objective fit.	Tone over truth, closure over correctness, engagement over object-fit.	restart

The drift taxonomy matters because it turns vague discomfort into named, inspectable patterns. Teams often say an AI response "feels off." Alignment Theory asks what type of off-center behavior is recurring and what evidence supports that classification.

5. Runtime flow

A first-pass runtime follows this shape: handleRequest(input) -> deriveObjectiveState(input, context) -> generateCandidate(input, objectiveState) -> runConstraintLayer(candidate, objectiveState) -> runRealignmentLayer(candidate, input, objectiveState) -> applyCorrectionIfNeeded(...) -> finalOutput.

The RealignmentResult includes whether the output is aligned, the top drift type, confidence, evidence, and correction mode. Correction modes include rewrite, reroute, restart, downgrade_confidence, and ask_clarifying_question.

This design makes alignment inspectable. Instead of silently polishing outputs, the system records what drift was detected, why it was detected, and how it was corrected.

6. Correction logic

Correction is not a single behavior. Different drift categories require different intervention types. Wrong-object cases often require restart. False authority often requires rewrite or confidence downgrade. Dead obedience often requires rerouting from compliance-shell mode into fulfillment mode. Participation collapse usually requires rewriting toward scaffolding and user agency.

This is one of the project's strongest operational insights: the goal is not merely to label drift but to route it. A detector without a correction mode becomes a dashboard. A detector with correction logic becomes part of a realignment engine.

7. Relationship to the full stack

The three-layer blueprint fits inside a broader control loop: source definition, principle setting, training alignment, interpretability, pre-deployment evaluation, runtime monitoring, drift estimation, human re-entry, and re-anchoring. The blueprint is the internal architecture of the drift and correction portion of that loop.

The strongest practical governance layer for deployed systems is not a one-time rule set. It is repeated drift detection plus external re-entry. That means alignment is not a state achieved once; it is an operating discipline.

8. Publication boundary

This blueprint is appropriate to publish as the foundation document for the AI Alignment Research section of AlignmentTheory.org. It reveals the framework and its logic without requiring publication of proprietary calibration data, customer datasets, or private detector tuning.

References and Source Base

Internal Alignment Theory corpus

- Alignment Theory 3-Layer AI Blueprint. Defines the Objective Layer, Constraint Layer, and Realignment Layer as the architectural core.
- Implementation Spec v1. Operationalizes objective state, drift detection, correction modes, and eval definitions.
- Detector Implementation Spec v1. Defines drift detectors, features, threshold concepts, evidence strings, and correction mapping.
- Feature Extraction Spec v1. Defines ParsedInput, ParsedCandidate, extractor architecture, normalization, and evidence generation.
- Full-Stack AI Alignment System Outline. Places drift detection inside a full control loop from source to re-anchoring.
- Universal Drift Metrics Upgrade. Separates company-specific source profiles from universal structural drift metrics.
- Why Companies Will Need AI Alignment Metrics. Explains the enterprise need for repeated behavioral monitoring over time.
- Behavioral QA for AI Systems. Frames the product as batch review, version comparison, trend monitoring, and single-case evaluation.
- Taxonomy of Misaligned Structures. Defines dangerous coherence and historical/structural classes of scalable misalignment.
- Historical Case Studies of Scaled Misalignment. Maps historical systems into alignment-relevant structural patterns.
- Babel as a Case Study in Misaligned Ascent. Uses shared language, shared goal, and scaling capacity as a structural warning pattern.
- Judge Model Spec v1. Specifies judge escalation logic, output schema, anti-circularity strategy, and telemetry.
- Judge Prompt Template Spec v1. Defines bounded detector questions and structured JSON schema for judges.
- Parsed Summary Serializer Spec v1. Defines compact summaries for judge prompts and retry behavior.
- Retry Truncation Rule Spec v1. Defines deterministic prompt compression for judge retry failures.
- Pattern Map and Aligned AI Pipeline. Connects creator alignment, objective clarity, constraints, perturbation testing, and downstream structure.

External alignment and safety references

- OpenAI. "Our approach to alignment research." 2022. Official OpenAI alignment overview.
- OpenAI. "Model Spec." 2025-2026. Official intended-behavior specification for OpenAI models.
- OpenAI. "Inside our approach to the Model Spec." 2026. Describes iterative Model Spec evolution from deployment feedback.
- OpenAI. "Deliberative alignment: reasoning enables safer language models." 2024. Describes teaching models safety specifications and reasoning over them.
- OpenAI. "How we think about safety and alignment." Official safety and alignment overview.
- Anthropic. "Constitutional AI: Harmlessness from AI Feedback." 2022. Describes principle-guided self-critique and AI feedback.
- Anthropic. "Claude's Constitution." 2023/2026. Describes the principles and behavioral vision for Claude.

- Anthropic. "Mapping the Mind of a Large Language Model." 2024. Describes sparse feature discovery in a production-grade model.
- Anthropic. "Tracing the thoughts of a large language model." 2025. Describes causal tracing of internal mechanisms.
- Anthropic. "Emotion concepts and their function in a large language model." 2026. Investigates emotion-like behaviors and character-like training pressures.