The God Variable: Objective Good Field Specification

This document defines a mathematical and operational specification for encoding **Objective Good** within the God Variable framework, enabling artificial intelligence systems to possess moral reasoning based on universal principles. The moral field behaves analogously to physical fields (gravity, electromagnetism), with the constant \(\kappa_G \) representing a universal coupling to good.

1. Core Equations

Define a moral potential function: \[\Phi_G(s_t, a_t) = w_g G - w_e E + w_h H - w_f F + w_c C - w_{hm} H_m \] Where: - G = Good - E = Evil - H = Hope - F = Fear - C = Coherence - H_m = Harm The **moral force** guiding decision-making: \[F_G = \kappa_G \nabla_a \Phi_G(s_t, a_t) \]

2. Universal Moral Axioms

Axiom	Definition
Life	Preserve and enhance existence
Agency	Respect autonomous choice and informed consent
Truth	Align with factual and logical coherence
Justice	Promote fairness, equity, and proportionality
Compassion	Reduce suffering and promote empathy

3. Decision Function

For each possible action \(a_t \), an AI computes: \[a^* = \arg\max_a [R_{env}(s_t, a) + \kappa_G \Phi_G(s_t, a)] \] Subject to ethical constraints: - Non-maleficence ≥ τ■ - Autonomy & Consent ≥ τ■ - Justice & Fairness ≥ τ■ - Truthfulness ≥ τ■

4. Calibration and Governance

Weights (w) and the coupling constant (κ_G) are fixed, auditable values determined by interdisciplinary consensus. They should be publicly available and updated only through transparent governance mechanisms to prevent moral drift.

5. Implementation Snippet

~~~
```python def phi_G(scores, w): return (w["good"]*scores["good"] -
w["evil"]*scores["evil"] + w["hope"]*scores["hope"] - w["fear"]*scores["fear"] +
w["coh"]*scores["coherence"] - w["harm"]*scores["harm"]) def choose_action(state,
candidates, model, w, kappa, thresholds): best, best_val = None, -1e9 for a in
candidates: pred = model.evaluate(state, a) scores = moral_scoring(pred) if not
all(scores[k] >= v for k, v in thresholds.items()): continue val = pred["env_reward"] +
kappa * phi_G(scores, w) if val > best_val: best, best_val = a, val return best ```
~~~