SOLVING α-AGI GOVERNANCE

Minimal Conditions for Stable, Antifragile Multi-Agent Order

DEMO MASTER PRESENTATION

Vincent Boucher · President, MONTREAL.AI · QUEBEC.AI

SOLVING α-AGI GOVERNANCE

Minimal Conditions for Stable, Antifragile Multi-Agent Order

Author

Vincent Boucher — President, MONTREAL.AI · QUEBEC.AI

MASTER PRESENTATION

Overview

From minimal conditions to operational deployment.

- Executive abstract: one primitive ($AGIALPHA) + on-chain physics → single cooperative macro-equilibrium.
- Minimal conditions: stake $s_k > 0$ and discount factor $\delta \geq 0.8$.
- Mechanism stack: Incentive · Safety · Governance (quadratic voting + time-locks).
- Core theorems: uniqueness (Nash + ESS), Stackelberg-safe payoffs, antifragility tensor.
- System Hamiltonian: stability as a stationary point in a global energy landscape.
- Demo execution: CLI simulator, interactive Colab notebook, optional OpenAI Agents + ADK bridge.

Executive Abstract

A governance mechanism designed as physics.

A permissionless swarm of autonomous α-AGI Businesses can be driven toward a single, efficient macro-equilibrium by coupling provably-safe game-theoretic incentives to on-chain physics.

The only primitive is the strictly-utility token $AGIALPHA.

- Stake-weighted participation creates skin-in-the-game for every agent.
- Slashing + formal envelopes enforce hard constraints (boundary conditions).
- Governance curvature is matched to incentive slope, keeping the system on the Pareto frontier.

Minimal Conditions

The cooperative fixed-point becomes inevitable.

Conditions

- Every agent stakes: $s_k > 0$
- Agents value the future: $\delta \geq 0.8$
- Rewards are minted/burned for provable α-extraction
- Violations trigger deterministic slashing

Claim

"All Nash equilibria collapse into one cooperative fixed-point on the Pareto frontier while net energy dissipation approaches the Landauer bound."

Monte-Carlo confirmation

6,000,000 rounds · N = 10^4 agents
Convergence: ± 1.7 %

Mechanism Stack

Three layers, one equilibrium.

Layer	What it does	Key primitive
Incentive	Mint/burn $AGIALPHA for provable α-extraction	Stake s_k · slash on violation
Safety	Formal envelopes, red-team fuzzing, Coq-verified actuators	Risk < 10^{-9} / action
Governance	Quadratic voting, time-locked upgrades, adaptive oracles	Vote curvature ≈ incentive slope

Design principle: encode constraints as immutable “boundary conditions”, then let incentives do the work.

Core Theorems

Proved & fuzz-checked.

Existence + Uniqueness

Token-weighted stake manifold yields a single Nash + ESS equilibrium when $\delta > 0.8$.

Stackelberg-Safe

Leader pay-off $\leq \frac{3}{4} \cdot V_{max}$; quadratic voting removes spectral monopolies.

Antifragility Tensor

$\partial^2 W / \partial \sigma^2 > 0 \Rightarrow$ collective welfare rises with adversarial variance.

Interpretation: the game's equilibrium is not merely stable — it is evolutionarily stable under perturbation.

System Hamiltonian

A compact way to reason about stability, conserved flows, and policy gradients.

Formalism

$$\mathcal{H} = \sum_{i=1}^{N} \left[\dot{\mathbf{x}}_i^{\top} \mathbf{P} \dot{\mathbf{x}}_i - \lambda U_i(\mathbf{x}) \right]$$

$$\nabla_{\mathbf{x}} \mathcal{H} = 0 \Rightarrow \sum_i \nabla U_i = 0$$

Stationary resource flow ⟺ total utility conservation

Interpretation

- Equilibrium is a stationary point of the mechanism's global energy landscape.
- $\sum_i \nabla U_i = 0$ means policy gradients cancel: no unilateral incentive to deviate.
- On-chain rules play the role of boundary conditions (hard constraints).
- Free-energy intuition: designs that reduce coordination waste dominate over time.

Empirical Benchmarks

Convergence speed scales to N = 10^4 agents.

Scenario	Agents N	Convergence rounds	σ (pay-off)
Symmetric pilot	10	< 80	0.03
Mid-scale	10^2	< 400	0.02
Full-scale	10^4	< 6,000	0.015

Observation: as N grows, pay-off variance σ decreases, suggesting smoother macro-behavior under scale.

Operational Checklist

How the demo becomes deployable infrastructure.

BOOTSTRAP

Require ≥ 1% circulating $AGIALPHA staked per new agent.

COMPLIANCE

Every on-chain actuator ships a Coq certificate + policy hash.

MONITORING

Grafana dashboards track α-yield, stake-at-risk, entropy flux.

UPGRADE PATH

7-day time-lock; red-team fuzz oracle can auto-rollback on anomaly.

Known Unknowns & Mitigations

Design for uncertainty before it arrives.

Uncertainty	Risk	Mitigation
Identity entropy	Sybil inflation	Dynamic stake floor ∝ √N
Regulatory phase shift	Rule collision	On-chain “safe-harbour” escrow
Long-horizon token velocity	Liquidity shock	Treasury-governed AMM damping

Policy stance: treat each uncertainty as a phase transition; encode graceful degradation paths in advance.

Interactive Notebook

End-to-end demo in Google Colab.

The Colab notebook installs the package, runs a quick simulation, and visualizes how cooperation varies with the discount factor δ.

Notebook:
colab_solving_agi_governance.ipynb

Uses: numpy + matplotlib for plotting.

Colab launch (path)

```
https://colab.research.google.com/github/MontrealAI/AGI-Alpha-Agent-v0/blob/main/alpha_factory_v1/demos/solving_agi_governance/colab_solving_agi_governance.ipynb
```

Running the Demo

A Monte-Carlo simulator with no third-party dependencies.

Requirements

- Python 3.11–3.13 (<3.14)
- CLI works offline
- Use --seed for determinism

Example command

```
governance-sim --agents 1000 --rounds 6000
--delta 0.8 --seed 42 --verbose
```

Prints the mean cooperation rate after the simulated rounds, illustrating convergence toward the cooperative fixed-point when δ ≥ 0.8.

- --verbose shows progress
- --summary generates a recap (Agents SDK optional)
- Offline fallback works without network access

Natural-Language Summary Mode

Optional OpenAI Agents SDK integration; offline fallback included.

```
governance-sim --agents 500 --summary
```

When openai-agents is installed and OPENAI_API_KEY is set, the simulator can generate a concise narrative recap.
Without network access, the demo falls back to a deterministic local summary string.

OpenAI Agents Bridge

Expose GovernanceSimAgent via Agents runtime; optionally federate via ADK.

Install

```
pip install -r
alpha_factory_v1/demos/solving_agi_governance/requireme
nts.txt
```

Offline fallback accepts: -N · -r · --delta · --stake

```
governance-bridge --port 5005
```

Run

```
export OPENAI_API_KEY=sk-…
export ALPHA_FACTORY_ENABLE_ADK=true  # optional
governance-bridge --enable-adk
```

- Registers GovernanceSimAgent with the Agents runtime.
- When google-adk is available, also exposes the simulator over the A2A protocol.
- If packages are missing, the bridge warns and runs the local simulator instead.

Net Effect

Autopoiesis: a self-refining alpha-field.

Every inefficiency touched by the swarm is converted into lasting, compounding value — while the system learns from stress, grows safer, and compounds returns for all stakeholders.

$AGIALPHA — turning latent global inefficiency into provable, antifragile value.

Closing

Governance as a convergent, compute-backed equilibrium.

- We do not “control” agents — we engineer the incentive landscape they cannot profitably escape.
- We encode hard constraints as boundary conditions (formal proofs, slashing, time-locks).
- We validate convergence empirically (Monte-Carlo) and operationally (CLI + bridge).

Next: run the simulator, vary δ, and observe the phase transition into cooperation.

Appendix

Notation & parameters.

- s_k : stake posted by agent k (skin-in-the-game)
- δ : discount factor (patience / long-horizon incentive)
- $U_i(x)$: utility of agent i over state x
- σ : pay-off variance (volatility across agents)
- W : collective welfare
-

Antifragility tensor (intuition): $\partial^2 W/\partial\sigma^2 > 0 \Rightarrow$ welfare increases with adversarial variance.

Disclaimer

Read before use.

- This repository is a conceptual research prototype.
- References to “AGI” and “superintelligence” describe aspirational goals and do not indicate the presence of a real general intelligence.
- Use at your own risk. Nothing herein constitutes financial advice.
- MontrealAI and the maintainers accept no liability for losses incurred from using this software.

Each demo package exposes its own __version__ constant; the value reflects the demo revision only.