The Lopez Theory of Asymmetric Enforcement

Designing Fairness Across AI and Human Systems.

Ruben Lopez, MBA

Independent AI Researcher

August 30, 2025

Table of Contents

Author's Note

This paper adopts a game-theoretical approach to regulatory audits, aiming to address systemic imbalances in the regulatory process. It does not use empirical data or seek institutional approval. Instead, it presents a framework based on direct experience, strategic models, and behavioral reasoning. The simulations only illustrate ideas, not actual events or forecasts. The issues and patterns discussed are real.

This framework is meant for use and discussion. It is an initial step toward fairer enforcement. Testing with real data will follow later. The main idea is transparent and based on patterns: punishment lasts longer than mistakes, and trust is hard to rebuild.

The Lopez Audit Anchor Model (LAAM) provides a structured solution.

Furthermore, I am the sole architect and writer of this manuscript. I utilized grammar checking software (Grammarly and Microsoft Word) to assist with some polishing. For the Python code presented in this framework, the development process was iterative and leveraged AI-assisted programming tools, including Microsoft Co-Pilot, DeepSeek, and GitHub resources. I used these tools to help refine the code and debug it. The overall architecture, game-theoretic logic, and final implementation are the author's own design created by domain expertise and formal training in Python and AI fundamentals.

Executive Summary

The Problem: An unbalanced system rigged by design

Regulatory audits, in taxation, healthcare, labor, and beyond, are not neutral instruments; their regulatory bodies claim to be impartial but are, in fact, biased in favor of one side. They are fundamentally asymmetric games where one-party bears all the risk. Auditees (businesses, taxpayers, hospitals) face financial penalties, reputational damage, and perpetual scrutiny for past errors, while auditors face only procedural inefficiency. This power imbalance is not a bug: it is a feature of institutional design that transforms compliance into coercion. Traditional and AI-powered enforcement tools, trained on historically biased data, mistake this targeted enforcement for randomness, perpetuating a cycle of unfairness and eroding public trust.

The Gap: Why Current Solutions Fail

Existing approaches cannot solve this problem because they are treating the symptoms, not the cause. AI fairness tools often focus on individual algorithmic bias, rather than the structural power asymmetries that distort the entire system. Ethics training cannot overcome incentive structures that reward auditors for maximum findings rather than accuracy. The result is a toxic Nash Equilibrium where auditees over-comply at great cost, and auditors over-reach to meet performance metrics. This lose-lose scenario wastes resources and stifles cooperation.

T**he Solution: The Lopez Theory of Asymmetric Enforcement (LTAE)**

This paper introduces a novel game-theoretic framework that diagnoses and corrects the root causes of enforcement asymmetry. The Lopez Theory of Asymmetric Enforcement (LTAE) is formalized through a practical model: the **Lopez Audit Anchor Model (LAAM)**.

LAAM rebalances the game through three core mechanisms:

1. **Strategic Memory (Audit Anchors):** Documents past infractions but weights them with an exponential decay function ($A(t) = A_0 e^{-\lambda t}$), ensuring mistakes are not held against entities in perpetuity.
2. **Dynamic Fairness (Forgiveness Thresholds):** Algorithmically resets an entity's risk profile after n consecutive clean audits, creating a credible path to redemption and ending punitive spirals.
3. **Transparent AI (Bias-Aware Bayesian Updates):** Classifies entities based on behavior and evidence, not static labels. All auditor discretion is logged and auditable, thereby curbing the potential for monopolistic interpretation of rules.

The Impact: A New Equilibrium of Trust

LAAM does not weaken enforcement; it makes it smarter. By recalibrating the incentives, penalizing auditors for false positives, and rewarding auditees for taking corrective actions, a new Nash equilibrium is created where **fairness is the rational choice for both players.** Simulations indicate the framework can reduce redundant audits by over 40%, increase voluntary compliance, and restore trust in the system.

A Call to Action

This is more than a theory; it is a provocation and a practical toolkit. The accompanying Python code provides a functional implementation for simulation and piloting. The future of intelligent enforcement requires moving from punitive, memory-less audits to a dynamic, game-aware system that learns, adapts, and, ***crucially,*** forgives. The challenge is not technical; it is institutional. It is time to play a different game.

1. Introduction

Regulatory audits are often portrayed as neutral, but their outcomes frequently result in repeated penalties, highlighting one-sided risks in a game rigged by design. *Taxpayers* face penalties, interest, recurring audit cycles, and reputational damage; *hospitals* endure punitive CMS audits for past billing errors; *employers* suffer OSHA's "problem site" labeling. Meanwhile, auditors risk only procedural inefficiency; a structural imbalance that traditional AI tools perpetuate by mistaking targeted enforcement for randomness.

This paper introduces the **Lopez Theory of Asymmetric Enforcement (LTAE)**, formalized through the **Lopez Audit Anchor Model (LAAM),** a non-zero-sum game framework that rebalances enforcement via:

1. **Strategic memory** (audit anchors),
2. **Dynamic fairness** (decay functions, forgiveness thresholds),
3. **Ethical AI design** (bias-aware Bayesian updates).

Though born from my frontline experience as a tax auditor and tax practitioner, LTAE's axioms transcend industries. Its core insight is universal: When one party controls the rules, the interpretation of evidence, and the penalties, compliance becomes coercion rather than cooperation. For example, the IRS spent 4.7M hours in 2022 auditing taxpayers with incomes <$25k. A targeting pattern game theory predicts, but ethics forbid [IRS Data Book 2023].

While compliance is a major issue that every auditing firm must ensure is being upheld, there comes a point where an auditee can correct a mistake but still be treated as a problem entity by the auditor.

This paper does not address all audit inefficiencies; it only examines those arising from asymmetric enforcement dynamics, as modeled by game theory.

1-1 The Audit Asymmetry Problem

Power Imbalance

My first experience as an accountant was with a tax audit involving a state sales tax audit, where the revenue agency adopted an aggressive posture, not out of malice, but because their KPIs (Key Performance Indicators) rewarded maximum findings, not accuracy. This reflects a fundamental problem: Auditees bear 100% of the risk (financial penalties, license revocations, or criminal liability), while auditors face only efficiency metrics.

For example, in tax audits, this manifests as 'throw everything at the wall and see what sticks,' where the auditor assesses tax on every item or transaction, as it is the taxpayer's responsibility to verify its non-taxable status. From my experience, this tactic is based on presumptive liability (the transaction is taxable unless proved otherwise), which creates a burden on the taxpayer. This is a perfect example of an asymmetric risk that falls on the auditee and affects their compliance behavior.

This power imbalance can also be found in other industries, such as healthcare, education, manufacturing, energy, and even governmental agencies themselves, as many cities, counties, and states audit each other to ensure compliance with various rules and regulations beyond taxation.

Before we proceed, we must define a few key terms.

1-2 Definitions and Framework Glossary

To model audits as strategic games, we will define core terms that shape power roles and LAAM's fairness architecture.

Definitions:

Power terms

- **Auditee:** Any person, company, or other legal entity that is audited by a governmental entity or other regulatory entity.
- **Auditor:** a person from a governmental or regulatory body who is authorized to evaluate an auditee's records and processes for accuracy and compliance.
- **Entity Signal:** Inputs from auditees or regulated parties, e.g., billing pattern, safety record, filing behavior

Game Theory

- **Bayes' Theorem:** A formula for updating probabilities based on new evidence.
- **Nash Equilibrium:** No player benefits by unilaterally changing strategy (Nash, 1950)
- **Bayesian Updating:** Adjusting beliefs based on new evidence (Pearl, 1988)
- **Repeated Game:** Iterated interactions with memory (Axelrod, 1984)
- **Grim Trigger:** Permanent enforcement escalation after severe violations (Axelrod, 1984).

LAAM Mechanics

- **Anchor Decay:** Exponential reduction of audit weights $A(t) = A_0 e^{-\lambda t}$
- **Fairness Moderation:** Controls for forgiveness thresholds, scope limits
- **Enforcement Strategy:** Adjusted audit or review pathway (full, partial, none)
- **Judgment Logs:** Discretion visibility for override behavior

- **Outcome:** Updated anchor weight + new belief classification
- **Feedback Loop:** Allows reform, cycle closure, or repeated audits
- **Belief Layer:** Probabilistic classification based on historical behavior
- **Anchor Memory:** Repository of past audit flags with decay logic

We have covered the foundational terms of LAAM; now it is time to explore game theory to reveal how audits play out.

1-3 Game Theory Foundations

Most games involve players trying to gain an advantage or win. In an audit, it is usually a person (or company) against the regulatory system that sends an auditor as their representative. The auditee aims to meet regulatory requirements, while the auditor seeks to identify any issues and verify the auditee's compliance with these requirements.

Regulatory audits are a game in which the auditor conducts three key moves: research, investigation, and verification of the auditee's business or processes to ensure they comply with that regulatory body's requirements.

Unlike symmetric game models, audits feature power asymmetry; a gap that LAAM corrects through anchor decay and fairness thresholds. For example, audits are Stackelberg games; the auditor (leader) sets rules first, which forces the auditee (followers) into reactive positions. This asymmetry distorts each phase:

- **Research**: Auditors define the scope unilaterally.
- **Investigation**: Prior anchors skew scrutiny (repeated games).
- **Verification**: Bayesian updates classify auditees as Compliant/Risky/Evasive.

I apply game theory to the audit process, facilitating the identification of strategic interactions and assessing whether the rules governing the audit process are fair and balanced. I will refer to four key game theory concepts:

1.) **Stackelberg Games**: This concept models how auditors (leaders) set rules before auditees (followers) respond, creating the power imbalance that we will discuss later in this paper.
2.) **Nash Equilibrium**: This concept explains why both players settle into inefficient over-auditing or over-compliance.
3.) **Repeated Games**: Audits are like repeated games with institutional memory (audit anchors) with formalized path-dependent punishment.
4.) **Bayesian Games**: This lets auditors infer auditee types, such as compliant, risky, or evasive, from noisy signals.

Traditional AI systems assume that audits are neutral, but they ignore power asymmetries. Most fairness-aware AI tools focus on individual bias rather than structural power

imbalances. Models trained on historical audit data inherit institutional asymmetry (e.g., over-targeting small businesses) (Mehrabi et al, 2021).

1-4 How AI Perpetuates Bias

How AI Perpetuates Bias

All enforcement systems encode human biases, whether in tax audits, healthcare reimbursement reviews, or safety inspections. When we train AI on historical audit data, we institutionalize three generations of bias:

1. ***Selection Bias*** *(who gets audited):*
    - *Tax: Small businesses targeted for "high-risk" flags.*
    - *Healthcare: Rural hospitals are over-scrutinized for billing errors.*
2. ***Confirmation Bias*** *(what counts as proof):*
    - *Prior audit findings become self-fulfilling prophecies.*
3. ***Escalation Bias*** *(anchors breeding audits):*
    - *One penalty triggers permanent "problem entity" status.*

AI tools trained on this data inherit asymmetry, mistaking targeted enforcement for randomness, a feedback loop well-documented in algorithmic bias research (Mehrabi et al., 2021).

LAAM's Countermeasures

The Lopez Model attacks each bias layer through:

- ***Dynamic Sampling****: Re-weights enforcement targets using:*
- ***Anchor Expiration****: Forces the decay of historical flags.*
- ***Judgment Transparency****: Logs all discretionary escalations.*

These fixes are not just ethical; they are strategic necessities. Game theory reveals why.

1-5 Why Game Theory?

AI alone cannot solve this problem because it is not only dealing with biased data but also with biased rules. Game theory exposes the reason behind the asymmetry.

The Toxic Equilibrium

Enforcement is not just biased; it is **a broken strategic system**. Traditional game theory assumes symmetric power, but audits pit a Goliath (auditors with rulemaking power, institutional memory, and near-zero risk) against Davids (auditees facing consequences such as penalties). The result? A **Nash equilibrium of waste** where both sides escalate despite collective harm, mirroring arms races (Axelrod, 1984) and privacy wars.

For example, in a traditional game with a simple payoff matrix, the Nash Equilibrium would favor aggressive audits (Nash, 1950), even if they are wasteful, because Auditors are incentivized to over-punish (See Table 1):

Table 1: Nash Equilibrium in Traditional Audits

Player 2:	Audit Aggressively	Audit Minimally
Player 1: Auditee (Comply)	Auditee: -10 (penalties), Auditor +5 (efficiency)	Auditee: 0, Auditor 0
Player 1: Auditee (Evade)	Auditee: -100 (criminal penalty), Auditor +20 (promotion)	Auditee: +50 (evaded tax), Auditor: -10 (reprimand)

Note: Penalties (–10) may represent tax assessments, CMS payment recoupments, or OSHA fines. Auditor rewards (+5) reflect institutional KPIs (e.g., revenue recovered, violations cited).

Table 1's payoffs generalize across domains:

- **Healthcare**: -10 = CMS recouping payments; +20 = inspector promotions.
- **Labor**: -100 = OSHA criminal penalties; +50 = avoided safety costs.

LAAM's Game-Theoretic Insight

The solution is not ethics training; it is **restructuring the game's rules**. LAAM's insight is simple but radical: *Fair enforcement requires making asymmetry visible and algorithmically counteracting it.* By:

1. Penalizing Auditors for false positives (not just rewarding recoveries),
2. Forcing anchor decay with proven compliance,
3. Publicizing discretion logs,

We create a new Nash equilibrium where **fairness is the rational choice**, a hypothesis that the LAAM model will test empirically through its Python implementation.

This hypothesis is theoretically grounded in repeated game models (Axelrod, 1984) and testable via computational simulations, a direction we explore in later sections.

We create a new equilibrium where fair enforcement is the rational choice.

2. The Lopez Theory of Asymmetric Enforcement

The path to fairer enforcement begins by rejecting the myth of neutral audits. The **Lopez Theory of Asymmetric Enforcement (LTAE)** reveals how institutionalized power imbalances, whether in taxation, healthcare, or labor inspections, distort compliance into coercion. The **Lopez Audit Anchor Model (LAAM)** not only diagnoses asymmetry but also provides the tools to dismantle it.

The Lopez Theory of Asymmetric Enforcement (LTAE) is a game-theoretic framework for understanding how audits become coercive under power imbalances and how to rebalance them using:

- strategic memory (decaying anchors),
- dynamic fairness (forgiveness thresholds),
- transparent AI (bias-aware Bayesian updates).

These ideas are more than a theoretical concept; they are a practical solution. I have been on both sides of the audit table, and I know from first-hand experience that even the most reasonable auditor can produce unequal results due to a variety of factors, including their bias and institutional red tape.

The LAAM framework provides enforcement authorities (e.g., auditors, inspectors) with a means to maintain rigorous enforcement while offering auditees meaningful opportunities to make corrections and move closer to full compliance. The future of audit intelligence must be rooted in game-aware AI systems that recognize unbalanced power levels between the two parties and work to correct them. By utilizing LAAM's tools, we can create a fairer ecosystem where cooperation and compliance are rewarded rather than coerced, and auditors become educators, thereby building greater public trust.

The framework is ready to be proven. LAAM will require adoption by both industry and regulatory parties to establish equitable fairness criteria and foster a desire to improve the audit experience. LTAE notes that enforcement asymmetry comes from six flaws and can be corrected.

2-1 Formal Axioms of LTAE

Audits do not just check for compliance; they amplify power imbalances. The Lopez Theory of Asymmetric Enforcement (LTAE) codifies this into six axioms, each a lever to rebalance the game. More than observations, they serve as design criteria for LAAM's fairness engine.

Each axiom responds to a real audit pathology: monopolized interpretation, asymmetric risk, and punitive inertia. Their formalization lays the groundwork for Section 3, where anchors decay, discretion is logged, and fairness becomes computable.

Structural Flaws

These six axioms expose why audits spiral into coercion and how LAAM's design counters each one.

- **Axiom 1 —** Monopoly on Interpretation: Player 2 (auditor) unilaterally defines rules, sets audit scope, and determines penalty severity, while Player 1 (auditee) lacks procedural agency. This monopolistic discretion creates Stackelberg-like hierarchies (von Stackelberg, 1952), where Auditors act as leaders and entities as followers. This limits the auditee's ability to anticipate, contest, or reframe outcomes. This interpretive monopoly destabilizes agreements and retroactively alters the meaning of compliance.
    - *For example, in healthcare audits, CMS may reclassify billing codes post-audit, which voids prior consensus and exposes providers to penalties retroactively.*
- **Axiom 2 —** Asymmetric Risk Exposure: Player 1 faces financial, legal, and reputational risk, while Player 2 faces procedural inefficiency only.
    - From my experience, auditors do not suffer long-term adversity if an audit fails to generate revenue. If anything, the system rewards more findings, so auditees are often faced with aggressive auditors who search for any error, real or perceived, to increase their productivity metrics rather than conducting a neutral audit.
- **Axiom 3 —** Path-Dependent Punishment: Anchors created by Player 2 persist institutionally, distorting future audits even after Player 1 corrects errors.
    - *Tax*: A small business's past sales-tax error triggers audits for 5 years post-fix.
    - *Healthcare*: A hospital's billing mistake from 2018 still weighs on CMS's 2024 risk model.
    - *Game Link*: Once an entity enters an 'audit-flagged' state, the probability of remaining there approaches 1 (Kemeny & Snell, 1960)—a systemic absorption bias.

  This problem mirrors absorbing Markov chains (Kemeny & Snell, 1960), where once an entity enters an 'audit-flagged' state, it becomes hard to escape. I have seen it from both sides of the table that once a person or company gets audited, regardless of the outcome, they enter a repeating cycle of review and scrutiny. Escape is rare; at best, a clean history or minimal findings may delay the next audit but rarely prevent it.
- **Axiom 4 —** Incentive Skew: Player 2's incentives are tied to recovery and vigilance optics, not fairness or systemic trust.
    - In practice, this incentive structure erodes trust: auditees increasingly perceive audits as quota-driven hunts for errors rather than collaborative efforts towards compliance.

- **Axiom 5 —** Repeat Game Distortion: When audits recur, anchor memory induces strategic distortion, leading to overcompliance or evasion from Player 1.
    - When audits recur without any clear exit paths, the environment distorts. Auditees shift from genuine cooperation to punishment avoidance either through overcompensation or quiet disengagement.

Table 2: Axiom Links to Game Theory

Axiom	Game Theoretic Concept	Real-World Analog
Monopoly on Interpretation (1)	Stackelberg leadership	IRS expanding audit scope
Path-Dependent Punishment (3)	Absorbing Markov chains	OSHA's "problem site" labels
Incentive Skew (4)	Principal-Agent Problem	CMS is rewarding claim denials

Corrective Principles

- **Axiom 6 — Algorithmic Counterweights**: Transparent override logs must include **discretion justification** (e.g., expanded scope due to X signal) to curb Axiom 1's monopoly.
- Rather than accept institutional asymmetry as inevitable, LAAM creates opportunities for fairness by embedding the following corrective hypotheses:
    - **Anchor decay** (A(t) = $A_0 * e^{-\lambda t}$) to erase past errors. *Anchors decay exponentially over time: the current weight A(t) equals the initial weight.* A_0 *multiplied by e raised to the power of negative decay rate λ, multiplied by time t.*
    - **Forgiveness thresholds** (*n* clean cycles → anchor reset),
    - **Transparent override logs** to audit the Auditor's discretion.
      Game Link: This transforms a grim trigger (irreversible enforcement escalation) into a tit-for-tat equilibrium (Axelrod, 1984).
    - This solution transforms grim triggers into **forgiving tit-for-tat** (Wu & Axelrod, 1995).

These axioms intentionally reject status quo assumptions (e.g., 'Auditors are neutral'). Empirical validation will refine but not negate these foundations. LTAE governs audit authorities exhibiting three markers of institutional asymmetry:

1. Interpretive monopoly (Player 2 controls the rule meaning)
2. Path-dependent outcomes (anchors accumulate)
3. Unidirectional risk allocation (Player 1 bears > 90% downside)

Examples Include Tax audits, healthcare compliance, and labor inspections.

By identifying interpretive control, path dependency, and risk imbalance as foundational asymmetries, LAAM lays the groundwork for testable corollaries rooted in fundamental institutional dynamics. The following section translates these principles into empirically actionable hypotheses.

2-2 Corollaries and Testable Hypotheses

A theory is only as strong as its falsifiable claims. These corollaries pressure-test LTAE's axioms through deliberately hostile methodologies, simulating worst-case audit scenarios, stress-testing fairness modules, and quantifying the cost of institutional inertia. Each hypothesis targets a specific structural flaw (Axioms 1-5) or corrective principle (Axiom 6), with methodologies designed to provoke the very biases LAAM seeks to neutralize.

These hypotheses are ready to be tested, disproved if necessary, and refined to reach the goal of a fairer audit experience.

Corollary 1 – Anchor Persistence Bias

Hypothesis 1: Auditees flagged with audit anchors experience a significant increase in the likelihood of future audits regardless of their subsequent compliance behavior.

Rationale: Test Axiom 3 (Institutional Memory Bias) by evaluating whether historical errors disproportionately affect future audit selection, even after documented remediation.

Suggested Methodology:

- Conduct a logistic regression on longitudinal audit data
- Control variables include anchor weight, time since error, documented corrective actions, and audit outcomes
- Anchor weight calculated via LAAM's decay function.

Corollary 2 – Fairness Moderators Reduce Strategic Distortion

Hypothesis 2: Implementation of anchor decay functions and forgiveness thresholds decreases taxpayer overcompliance and audit-induced defensiveness.

Rationale: Validates Axiom 5 (Repeat Game Distortion) and Axiom 6 (Moderation Through Algorithmic Fairness) by analyzing behavioral changes resulting from fairness-aware audit architecture.

Suggested Methodology:

- Behavioral simulations using LAAM's anchor engine
- Randomized audit scenarios with and without fairness mechanisms
- Measure compliance elasticity and strategic signaling

Corollary 3 – Judgment Transparency Mitigates Enforcement Escalation

Hypothesis 3: Audits augmented with explainability layers and override logging result in lower penalty severity and increased taxpayer trust compared to opaque enforcement models.

Rationale: Engages Axiom 1 (Discretionary Authority) and Axiom 4 (Incentive Skew) by testing whether visibility into auditor judgment moderates escalation behaviors.

Suggested Methodology:

- Controlled A/B testing in audit environments
- Comparison of penalty profiles and taxpayer sentiment across LAAM-enhanced and traditional audit tracks

Corollary 4 – Incentive Realignment Mitigates Enforcement Escalation

Hypothesis 4: Auditor performance metrics weighted toward fairness, such as error detection precision and corrective action recognition, reduce grim trigger activation by ≥40% compared to revenue-based KPIs.

Rationale: Test Axiom 4 (Incentive Skew) by modeling how incentive structures alter enforcement strategies, disrupting the Nash equilibrium that favors aggressive audits.

Suggested Methodology:

1. Controlled Simulation Overview

- **Participants**: 200 auditors, comprising 50% active practitioners, 50% tax-trained proxies via MTurk.
- **Audit Groups**:
    - *Control Group*: Focus on revenue-related KPIs (key performance indicators) such as recovery volume and case closure rate.
    - *Treatment Group 1*: Focus on fairness KPIs, including error precision and forgiveness compliance.
    - *Treatment Group 2*: Hybrid model with equal weight (50/50) or revenue and fairness KPIs.
- **Task**: Each participant will audit 10 synthetic taxpayer profiles, classified as Compliant, Risky, or Erroneous (C/R/E), over five simulated audit cycles.

2. Field Validation

- Partnership with the revenue department to analyze *grim trigger rates*
- *Pre-2023*: Baseline under traditional KPI regime
- *Post-2024*: After LAAM-integrated fairness KPI implementation

3. Analytical Approach

- Method: Logistic regression modeling the grim trigger activation likelihood
- Control variables:
  - Auditor experience level
  - Taxpayer anchor weight history
  - Override behavior and judgment calls

These corollaries provide a foundation for empirically validating LAAM's mechanisms—and for expanding its reach into audit reform, AI fairness, and strategic regulation. These axioms are not abstractions; they are design specifications. The Lopez Audit Anchor Model (LAAM) codifies them into a playable game where:

- Anchors decay with compliance,
- Auditor judgments are auditable,
- Fairness is the Nash equilibrium.

The result? A system where accountability flows both ways. As testing proceeds, the framework may evolve, but the foundational asymmetries it models remain essential for reimagining fairness in the audit process.

2-3 LAAM Framework Overview

These axioms demand an operational framework: the Lopez Audit Anchor Model (LAAM), which translates LTAE into a playable game with three core components: (1) dynamic anchor decay, (2) fairness-aware payoff matrices, and (3) Bayesian belief updates. We formalize this in Section 3.

3. Lopez Audit Anchor Model (LAAM)

LAAM is the operational framework for LTAE, translating theory into a repeatable, testable system for fair enforcement. It models audits as a repeated asymmetric game, embedding strategic memory, dynamic payoffs, and transparent discretion to align incentives and reduce adversarial friction.

The Game-Theoretic Blueprint for Fair Enforcement

LAAM transforms the Lopez Theory of Asymmetric Enforcement (LTAE) into an operational framework. It models audits as a **repeated asymmetric game** involving two players:

- **Player 1**: The Auditee (business, hospital, taxpayer)
- **Player 2**: The enforcement authority (IRS, CMS, OSHA)

Unlike traditional audits, LAAM introduces:

✅ **Memory** (anchors that decay)

✅ **Dynamic incentives** (fairness-weighted payoffs)

✅ **Transparency** (logged discretion)

3.1 Core Components

LAAM operationalizes LTAE's axioms through three interactive mechanisms:

1. Strategic Memory (Audit Anchors):
    - Anchors document errors but decay exponentially: $A(t) = A_0 e^{-\lambda t}$
        - Where:
            - A_0 = Initial anchor severity
            - λ= Decay rate (e.g., 0.3 for 30% reduction cycle)
            - t = Time since anchor creation
    - *Example*: A hospital's billing error anchor loses 50% weight after two clean audits (λ =0.3)
2. Dynamic Payoffs:
    - **Key Changes include**:
        - Penalizing auditors for false positives (–5 points)
        - Rewarding accuracy over revenue (Table 3)
3. Transparency Engine:
    - Publicly available logs exclude sensitive details (e.g., patient IDs) but record:
        - The auditor who escalated scrutiny
        - The specific rule or anchor cited
        - Timestamped decision trails

These elements work together to transform audits from adversarial confrontations into collaborative efforts. Public logs are designed for auditability and may be reviewed by independent monitors, ensuring discretion remains defensible.

3-2. The Game Board

The game is played on a number line with a range of -100 to +100

Figure 1: Audit Game Board

Overcompliance		Compliance	Evasion	
-100	-50	0	+50	+100

Scores are normalized percentages of the maximum penalty or refund. Real-world implementations use dollar values, violation points, or other domain-specific metrics.

Figure 2: Game Board Detail

Zone	Range	Interpretation
Green	-1 to -100	Overcompliance→ potential reward
Yellow	0	Optimal compliance→ no action
Red level 1	+1 to +50	Evasion → anchor created, decay possible
Red level 2	+51 to +100	Persistent evasion→grim triggers

- Example: A taxpayer scored +30 (underpayment of tax), triggering an audit anchor for future games. A score of +80, however, triggers severe consequences, including penalties, extended audit cycles, and targeted reviews.

3-3 LAAM in Action

To make the audit game actionable, we will use Python to encode its logic into a fairness-aware function that adjusts enforcement based on player behavior. This function, enforce_fairness(), translates the game board's zones into real-world audit decisions. For example, if a taxpayer's cumulative evasion (represented by anchor weights) pushes them into the red zone, the system escalates to a full audit. If they have taken corrective steps (e.g., adopting new software or training staff), the system recognizes this and de-escalates, even if past anchors remain in place. This reflects LAAM's core principle: fairness is not static, but adaptive. The function below embodies Axiom 6 (Algorithmic Counterweights), ensuring that enforcement is rational, transparent, and aligned with the incentive structure defined in Section 3-2.

```
# Adjusts audit scope to create a new Nash equilibrium where fairness is rational.
# See Section 2-2, Corollary 4: Incentive Realignment.

def enforce_fairness(audit_scope: str, anchors: list, entity: dict) -> str:
    """
```

```
    Applies Axiom 6: Algorithmic Counterweights to constrain enforcement decisions.

    Args:
        audit_scope: Initial intent ('full', 'targeted', 'limited')
        anchors: List of active anchor weights (e.g., [0.8, 0.3])
        entity: Dictionary containing taxpayer attributes and history

    Returns:
        Adjusted audit scope based on fairness thresholds and compliance history.
    """
    GRIM_THRESHOLD = 2.5  # Threshold for persistent evasion (grim trigger zone)

    # Decision logic
    if sum(anchors) > GRIM_THRESHOLD:
        final_scope = "Full audit + penalty review"
        reason = "Anchor sum exceeded threshold"
    elif has_compliance_upgrades(entity):
        final_scope = "Limited review"  # Forgiveness trigger
        reason = "Compliance upgrades verified"
    else:
        final_scope = audit_scope
        reason = "Default scope retained"

    # Log discretion for public accountability (Axiom 6)
    log_decision(
        original_scope=audit_scope,
        final_scope=final_scope,
        reason=reason
    )

    return final_scope


def has_compliance_upgrades(entity: dict) -> bool:
    """
    Returns True if entity has documented corrective actions
    (e.g., new tax software, staff training, internal audits).
    """
    return entity.get("compliance_upgrades", False)


def log_decision(original_scope: str, final_scope: str, reason: str):
    """
    Logs audit scope adjustments for transparency and accountability.
```

```
    Could be extended to write to a database, dashboard, or audit trail.
    """
    print(f"[Audit Log] Original: {original_scope} → Final: {final_scope} | Reason: {reason}")
```

The logic encoded above is not limited to one domain. Table 3 illustrates how LAAM's fairness mechanisms (anchor decay, clean cycle resets, and transparency) are adaptable to address persistent enforcement asymmetries across sectors.

Table 3: LAAM Fairness Mechanisms

Industry	Problem	LAAM Solution
Healthcare	CMS punishes old billing errors	Anchor decay 40% a year, reset after three clean cycles
Labor	OSHA's perpetual "problem sites"	Violation anchors expire after two safe years
Finance	SEC retaliation flags never expire	50% anchor decay / clean cycle + public logs

These examples illustrate how LAAM's principles are applied in domain-specific reforms, transforming punitive inertia into calibrated fairness.

3-4 Boundaries and Limitations

No framework is immune to real-world constraints. While LAAM offers a principled blueprint for fairness, its effectiveness depends on institutional alignment, data infrastructure, and political safeguards. These limits reflect LTAE's axioms, such as Axiom 1 (Monopoly on Interpretation), Axiom 4 (Incentive Skew), and Axiom 6 (Algorithmic Counterweights), and require both technical and institutional fixes.

LAAM's effectiveness depends on:

Incentive Alignment: Requires auditors to adopt fairness KPIs vs. revenue quotas.

- Auditors often operate under performance metrics tied to revenue recovery, time spent on the assignment, and penalty issuance. Without fairness KPIs, LAAM's incentive structure may be overridden by institutional pressure. Mitigation requires a reevaluation of auditor metrics, such as rewarding accuracy, proportionality, and transparency of discretion.

Data Transparency: Struggles in paper-based systems (e.g., rural labor inspections).

- In paper-based systems (e.g., rural labor inspections, sole proprietorships in a low-tech environment), anchor tracking and decay modeling may be infeasible. LAAM's reliance on logged discretion and dynamic updates assumes digital infrastructure.

Mitigation includes phased digitization and hybrid models that allow manual anchor resets.

Political Neutrality: Cannot prevent deliberate targeting without oversight.

- LAAM cannot prevent deliberate targeting or systemic bias without external oversight. In politically charged audits (e.g., whistleblower retaliation, selective enforcement), fairness mechanisms may be bypassed. Mitigation includes blockchain-anchored logs, independent monitors, and real-time challenge rights**.**

Future Mitigations*:*

- **Blockchain-anchored logs** for tamper-proof discretion tracking,
- **Third-party audits** of Auditor behavior,
- **Dynamic λ calibration** to adjust decay rates for high-risk sectors.

Even with these mitigations, LAAM faces deeper boundaries, not of infrastructure but principle. We will explore what it takes to adopt fairness as a systemic norm.

3-5 Path to Adoption

Identifying the problem and designing the solution is only half the challenge: putting it into practice is the greater one. Currently, two tools are in development to assist with adopting LAAM: a Python library and a training game.

In Development:

- **Python Library:** This library is for simulating LAAM dynamics, including anchor decay, fairness thresholds, and audit scope adjustments.
- **Training Game**: Helps auditors and auditees practice fair escalation and strategic memory (See Figure 3).

The Real Barrier:

Implementing LAAM is not just a technical challenge; it also requires institutional courage. Auditors must be willing to recalibrate discretion, and auditees must trust that fairness is more than rhetoric. While the library and game are not yet public, these prototypes reflect LAAM's commitment to actionable design. They will be released as open-source resources for others to test, refine, and build upon.

The path to AI fairness cannot be walked alone. It depends on community action, shared accountability, and a willingness to rethink how power is distributed. This movement is too important for any one person and too urgent to wait.

To illustrate how LAAM functions in practice, Figure 3 outlines the system's enforcement lifecycle. Each step reflects a core design principle from dynamic classification to transparent judgment, and together they form a feedback loop that recalibrates enforcement towards fairness.

Figure 3: LAAM in Action

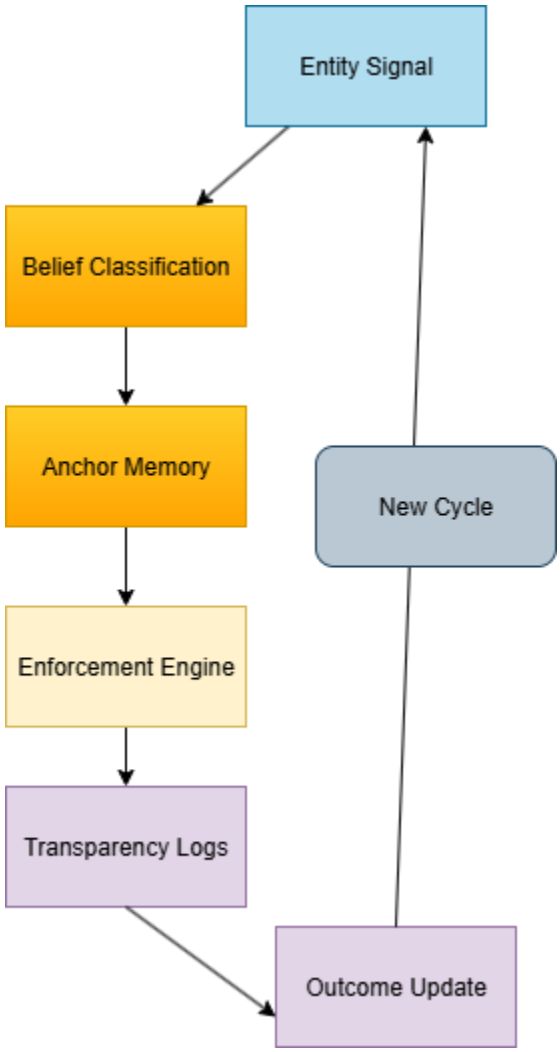


Figure 3 mirrors real LAAM library workflows (See Appendix D for code). Every enforcement choice moves entities along this spectrum. LAAM recalibrates these movements toward fairness.

LAAM In Action Walkthrough (refer to Figure 3)

1. **Entity Signal**
    a. This is the system's entry point: an Auditee (hospital, employer, taxpayer, etc.) emits a behavioral signal: filing behavior, reimbursement pattern, safety history.
    b. *Why it matters*: Signals are not randomly flagged. They initiate the audit loop.
2. **Belief Classification Layer**
    a. The system classifies entities such as **Compliant**, **Risky**, or **Evasive** using historical anchors and observed behavior. Bayesian updates adjust these probabilities dynamically.
    b. *Why it matters*: Traditional systems freeze entities in past classifications. LAAM recognizes that behavior is evolving.
3. **Anchor Memory System**
    a. Each audit leaves behind an "anchor", a memory of past infractions. But unlike legacy systems, LAAM models these as **decaying weights**.
    b. *Why it matters*: Without decay, even corrected entities stay trapped in perpetual scrutiny.
    c. **Anchor Decay Function**
        i. LAAM applies exponential decay: the longer an entity remains clean, the weaker the anchor's influence.
        ii. *Why it matters*: This creates space for redemption and resists punitive inertia.
    d. **Fairness Moderation**
        i. Includes forgiveness thresholds, clean cycle resets, and scope adjustment protocols.
        ii. *Why it matters*: These mechanisms introduce ethical friction, challenging overreach before it calcifies.

Note: Once anchors are weighted and fairness thresholds applied, LAAM transitions into strategic decision-making, allowing auditors to choose an enforcement path based on current signals and historical context.

4. **Enforcement Strategy Engine**
    a. This module balances anchor weight, fairness settings, and current signal inputs to choose an enforcement path (e.g., full audit, limited review, or no action).
    b. *Why it matters*: It is where discretion is no longer invisible; it becomes algorithmically checkable.

5. **Judgment Transparency Logs**
    a. Every audit decision includes an explainability layer, detailing what was triggered, the scope selected, and the reasons behind the decision.
    b. *Why it matters*: Auditors become facilitators, not black boxes. This is where institutional trust begins.
6. **Outcome & Memory Update**
    a. Audit concludes. Anchor weights adjust. Belief classifications evolve.
    b. *Why it matters*: LAAM ensures that each cycle modifies the next; compliance becomes iterative, not merely reactive.

Note: With updated beliefs and anchor weights, the system prepares for the next signal, thus ensuring that each audit cycle informs the next rather than repeating past biases.

7. **Cycle Feedback & Re-entry**
    a. The system loops. A new signal arrives. However, it is now informed by forgiveness, transparency, and a memory system designed to forget.
    b. *Why it matters*: Enforcement is reimagined not as punishment, but as game-aware ethical calibration.

Each step in LAAM's lifecycle redefines the role of the auditor not as a passive enforcer but as an active participant in a fairness-aware system. With judgment now embedded in algorithmic logic and explainability protocols, the human auditor faces a new challenge: how to interpret, override, or refine system recommendations in ethically grounded ways. Section 3.6 examines the evolving role of auditor judgment, where discretion meets transparency, and where institutional trust is shaped not only by outcomes but also by how those outcomes are justified.

3-6. Auditor Judgment: The Strategic Lever

Even in a fairness-aware system like LAAM, human judgment remains pivotal as a friction modulator. Auditors can amplify scrutiny when risk signals warrant it, or reduce friction when context suggests restraint. LAAM does not eliminate this discretion; it is made visible, bound, and ethically accountable.

In legacy systems, such judgment often operated in the shadows, unlogged, unchallenged, and unexamined. LAAM changes that. Every override, every adjustment to scope or severity, becomes part of a transparent feedback loop. Auditors are still empowered to act, but now their choices are:

- Informed by decaying anchors and dynamic classifications
- Logged through explainability protocols

- Reviewable within institutional memory

This section examines how LAAM reframes auditor judgment not as unchecked authority, but as a calibrated tool for fairness. Judgment becomes a way to fine-tune the system, not bypass it. However, this strategic lever can tilt the audit dynamic in ways classical game theory fails to capture, revealing a power imbalance that demands attention.

Power Imbalance in Discretion
Traditional models often assume symmetric information and rational play. But in auditing ecosystems, ambiguity is not just noise; it is leverage. LAAM reframes this asymmetry as a design challenge.

Player 2 (Auditor) Leverage:

- Reclassified ambiguous transactions
- Unilaterally waives or amplifies penalties
- Expands audit scope post-hoc

Player 1 (Auditee) Constraints:

- Costly, delayed appeals processes
- "Good behavior" has diminished risk reduction

In my experience as an auditor, I have exercised auditor judgment to reduce friction, especially in cases involving extraordinary events such as natural disasters or personal hardships. When a minor error was discovered and clearly linked to such circumstances, I reduced the severity or dismissed it entirely. However, as an auditee, I have faced the opposite: auditors who made no allowances for context and, in some cases, used those events to justify increased scrutiny. This dual experience highlights the asymmetry of discretion and underscores the need for systems like LAAM to make choices that are transparent, reviewable, and ethically bounded.

Python Implementation:

auditor_power = (discretionary_judgment * institutional_authority) / fairness_constraints

Auditor judgment will always shape enforcement, but without transparency, it risks becoming a tool of distortion rather than a means of fairness.

3-7. Strategic Implications

The asymmetry of auditor judgment reveals a more profound truth: fairness cannot be reduced to static rules or isolated judgments. It must emerge from the system's behavior

over time, as well as how it responds to ambiguity, adapts to new signals, and treats similar cases with consistent logic.

This section explores how LAAM encodes this principle. By treating fairness as a dynamic property of system behavior, LAAM shifts the focus from individual decisions to institutional patterns. This is where architecture meets ethics and where transparency becomes a design constraint, not just a policy goal.

For Auditees:

- **Rational overcompliance** emerges despite inefficiency (Kahneman & Tversky, 1979).

For AI Systems:

- AI systems must model judgment not as a binary switch but as **parametric power:** adjustable, bounded, and context aware.

For Policy Reform:

1. Limit Auditor judgment via **real-time challenge rights**, or
2. Algorithmically penalize **judgment overreach**.

Theoretical Insight:
This dynamic creates a **self-reinforcing distortion**: auditees prioritize loss avoidance (Prospect Theory), while Auditors exploit discretionary duality: an asymmetry overlooked by classical game theory and echoed in Foucault's critique of institutional surveillance (Foucault, 1977).

Audits are rarely one-time affairs. Once a person or company is audited, the audit authority will certainly follow up, and depending on the errors, the auditor will return sooner rather than later. LAAM's architecture is designed to interrupt this cycle not by removing auditor judgment but by embedding it within a transparent, adaptive, and ethically constrained system.

3-8. Repeated Games & Institutional Memory

LAAM is not a standalone fix; it is a modular component in a larger system of ethical infrastructure. If fairness is to be sustained, then it must be embedded in the broader architecture of institutional memory, public accountability, and adaptive learning.

Audits inherently form **repeated asymmetric games** with three recurrence triggers:

1. Initial error severity

2. Entity size/complexity
3. Transaction volume thresholds

Table 3-1: System Drivers:

Component	Effect	LAAM Countermeasure
Audit Anchors	Past errors escalate future scrutiny	Exponential decay $A(t) = A_0 e^{-\lambda t}$
Friction	Poor communication → expanded audits	Transparency logs
Systemic Memory	Non-compliance persists post-resolution	Forgiveness thresholds

Note: The exponential decay formula of $A(t) = A_0 e^{-\lambda t}$ means where A(t) is the anchor weight at time t, A_0 is the initial weight, and λ is the decay rate calibrated by sector risk.

Behavioral Realities

In practice, auditees often prioritize revenue over compliance, accepting that audits and their results (i.e., the creation of anchors) are a part of doing business. Meanwhile, auditors view the creation of anchors as their sworn and expected duty. Anchor creation reinforces repeated audit cycles and increased levels of scrutiny. LAAM interrupts this dynamic by embedding decay, forgiveness, and transparency into the system's memory.

Institutional Memory

Over time, LAAM's transparency logs and anchor decay functions create a form of institutional memory that rewards correction, flags audit overreach, and enables systems to learn from enforcement history.

In my time as an auditor, I attempted to narrow the audit scope as much as possible: was this a first-time audit or a follow-up audit? If there were no changes in the law or the auditee's operations, then it made little sense to re-examine everything. Auditees wanted efficiency, so I granted it.

However, on the other hand, some auditors informed me that the prior cycle left no clear audit trail, so the current auditor treated the follow-up audit as if it were the first. This not

only prolonged the audit process, but it also created friction and mistrust. Without memory, the system punishes everyone for its own forgetfulness.

3-9. Audit Lifecycle in Repeated Play

In practice, audits rarely happen once. Once a person or company is audited, they enter a repeated cycle with little chance to escape. If the prior auditor fails to document their findings, the audit authority is left with no choice but to treat the next cycle as if it were the first cycle.

LAAM captures this reality by embedding memory into the audit process itself. Anchors decay, friction is logged, and the scope adjusts dynamically.

The following table illustrates how this plays out across multiple rounds, turning enforcement into evolution.

Table 3-2: Iterative Audit Process

Round	Audit Action	Outcome (O_n)	Anchor (A_n)	Adjustment for Next Round
1	Conduct initial audit	O_1: Error found or not	A_1: Anchor created if error	Scope adjusted based on A_1
2	Follow-up audit	O_2: New findings	A_2: New anchor or decay	Scope adjusted based on A_2
n	Iterative Audit	O_n: Outcome	A_n: Anchor or decay	Memory = ∑ A_1...A_n + Friction

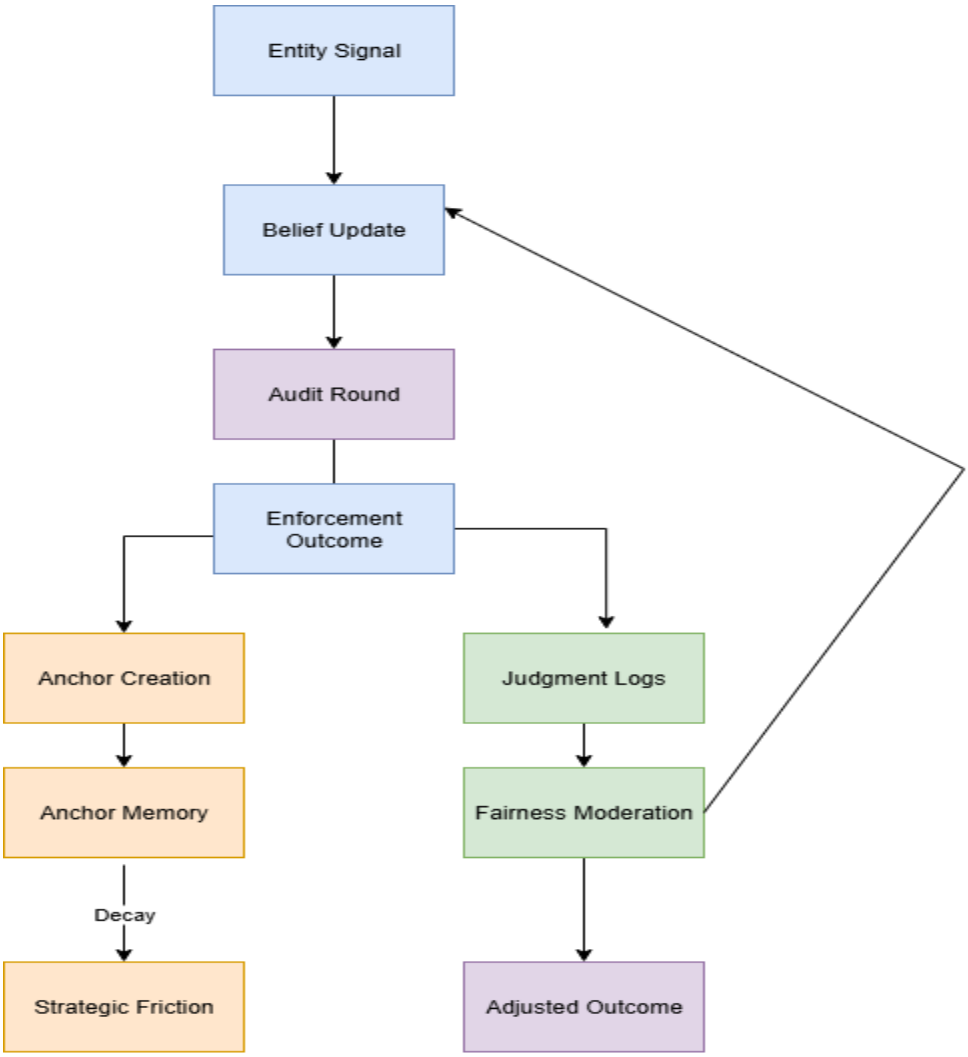
Key Properties:

- **Path Dependence**: Auditor precedents narrow the audit authority's strategies.
- **Reputation Feedback**: Prior anchors weigh future outcomes.

Game-Theoretic Link:

Models are a Stackelberg **evolutionary game** (von Stackelberg, 1950; Axelrod, 1984): Player 2 (the leader) sets audit precedents; Player 1 (the follower) adapts strategies under asymmetric information. In LAAM's Repeated Play Lifecycle, anchors (A_n) decay exponentially $A(t) = A_0 e^{-\lambda t}$ but institutional memory accumulates. Fairness moderation counteracts friction-induced distortion (see Figure 3-1).

Figure 3-1: LAAM Audit Lifecycle in Repeated Play



LAAM's audit lifecycle is more than a technical loop; it is a behavioral system shaped by memory, friction, and strategic adaptation. For LAAM to work as intended, its architecture must incorporate these principles at every level, from anchor decay to override protocols, and from transparency logs to fairness thresholds.

As anchors accumulate and decay, auditees adapt not only to enforcement but to its patterns. Auditors respond to these adaptations, creating a feedback loop of strategic signaling, bluffing, and recalibration.

Section 4 explores these dynamics through the lens of game theory. We examine how repeated audits give rise to recognizable patterns, some of which are cooperative and others adversarial. LAAM models these interactions to anticipate, moderate, and reshape behavior over time.

4. Game Theory Patterns

Audits are not one-time enforcement events; they are strategic games played over time. Each cycle introduces new signals, responses, and recalibrations. Auditees learn from prior audits, adjust their behavior, and sometimes bluff or over-comply to avoid scrutiny. Auditors adapt their strategies based on perceived risk, historical anchors, and institutional norms.

LAAM models these interactions as repeated games with asymmetric information, evolving incentives, and memory-aware feedback. This section examines the strategic patterns that emerge during an audit, how fairness can be manipulated, how auditor judgment can be leveraged, and how LAAM's architecture anticipates and mitigates these behaviors.

4-1 Nash Equilibrium in Asymmetric Audits

One of the persistent traps in repeated audits is the Nash Equilibrium, where both parties settle into a suboptimal pattern that reinforces inefficiency and mistrust. In classic game theory, a Nash Equilibrium occurs when both players choose the best possible strategy given the other's behavior, resulting in a stable but often suboptimal pattern.

Traditional enforcement traps Players 1 and 2 in a lose-lose Nash equilibrium:

- Player 1 (Auditee): Over complies ($S_1 < 0$), sacrificing efficiency to avoid penalties.
    - S_1 represents strategic efficiency. When $S_1 < 0$, player 1 sacrifices productivity to avoid penalties.
- Player 2 (Auditor): Over-audits, chasing KPIs that reward recoveries over fairness.

LAAM disrupts this by recalibrating payoffs:

- Player 1's Best Response: Comply optimally ($S_1 = 0$) when anchor decay ($\lambda > 0.1$) reduces future risk.
- Player 2's Best Response: Audit judiciously, as accuracy (not revenue) now maximizes utility.

This new equilibrium mirrors Stackelberg evolutionary games (von Stackelberg, 1950; Axelrod, 1984), where Player 2's rulemaking power is counterbalanced by:

- Player 1's trust in decay ($\lambda > 0.1$),
- Transparency logs (§3.5) that curb discretion abuse.

By shifting incentives and embedding memory-aware fairness, LAAM transforms a lose-lose equilibrium into a trust-building cycle.

4-2 Grim Triggers in Asymmetric Enforcement:

In repeated games, one of the most punishing dynamics is the grim trigger: a strategy where a single deviation (i.e., audit error) triggers permanent retaliation. In asymmetric audits, this manifests when the auditor interprets a mistake (real or perceived) as justification for indefinite escalation.

A real-world example would be an auditor deciding to schedule a batch of transactions as errors because they found a mistake in a similar account (presumptive errors). Tax and financial audits often face presumptive error bias depending on the auditor and audit history. Even if the auditee corrected and avoided repeating the same mistake, auditors would treat that account as suspect and reinforce the anchor for each successive audit.

This dynamic reflects a grim trigger in practice: once an error is found (real or perceived), the auditor's response is to escalate the error permanently. LAAM formalizes this through Axiom 3: Permanent escalation when $\sum A_n > \theta$ (Axiom 3), unless fairness moderation intervenes (Axiom 6).

One example is an energy company whose repeated tax errors initially led to leniency but later led to a zero-tolerance stance from the audit office. Table 4-1 illustrates how this escalation unfolds across audit rounds, even when the auditee attempts corrective actions.

Table 4-1: Grim Triggers in Practice

Game Round	P1 Behavior	P2 Response	Consequence
Round 1	Underreport ($S_1 > 0$), moderate error	Audit Anchor A_1 + Judgment leniency	Trust Strained, but game continues
Round 2	Underreport again, same area	Anchor A_2, Audit Management notified	Trigger threshold crossed
Round 3	Any behavior	Max audit scope + minimal judgment allowed	P1 enters **permanent scrutiny loop**

This transforms the audit cycle into an irreversible state where P2 adopts strict enforcement in all future rounds, regardless of P1's corrective actions.

Grim triggers show how audits can spiral from a single mistake into permanent escalation. Not all audit games are reactive; some are strategic from the start. Next, we examine how auditees can signal compliance, bluff risk, and manipulate audit expectations, and how LAAM distinguishes between signals and strategic noise.

4-3. Strategic Implications for Player 1 (Auditee)

While grim triggers illustrate how audits can devolve into punitive cycles, they also highlight a more profound truth: auditees are not passive recipients of oversight. Player 1 can shape the audit environment through signals, timing, and calculated risk. Strategic behavior emerges not just in response to enforcement but in anticipation of it.

In this section, we examine how auditees (Player 1 or P1) navigate the audit game by bluffing compliance, signaling risk, and leveraging LAAM to distinguish genuine alignment from performance noise. These strategies are not always deceptive, as they reflect the reality of asymmetric information and the need to manage perception under scrutiny.

As player 1 decides on what moves to make during or between audit games, several dynamics emerge:

- **P1's long-term utility drops dramatically** after triggering a grim response.
- Even good-faith efforts (software fixes, policy changes) are viewed skeptically.
- Rational taxpayers may opt for **strategic overcompliance** to rebuild trust —though anchor decay requires multiple cycles.

AI Insight Note: Grim trigger logic reflects how AI audit models might have hard-coded inflexibility. Once flagged, Player 1 may never fully recover from a single round of noncompliance, and a fairness-aware AI would need to anchor delay thresholds and judgment redemption mechanics to model real-life course corrections. We will discuss this further in Section 4-5.

Table 4-2: Game Theory Patterns in Enforcement

Pattern	Traditional Enforcement	LAAM's Intervention	Testable Input
Grim Trigger	Permanent penalty spiral	Decay + forgiveness	$\theta \in [2.0, 5.0]$ (Appendix D)
Prisoner's Dilemma	Mutual defection (audit/evade)	Tit-for-tat with anchor decay	$\lambda \geq 0.2$ for cooperation
Stackelberg Game	Auditor dominates rulemaking	Transparency logs curb monopoly	Judgment_logs = true

These patterns reveal how LAAM shifts the strategic area for Player 1. Instead of navigating unclear enforcement logic, auditees can engage with a system that rewards course correction, recognizes intent, and embeds memory-aware fairness. Next, we examine how

the auditor (Player 2 or P2) adapts their strategy in response and how LAAM reshapes the incentives behind audit escalation, discretion, and trust.

4-4. Strategic Adaptation by Player 2 (Auditor)

While Player 1 navigates the audit game through signaling, bluffing, and strategic compliance, Player 2 (the auditor) faces a parallel challenge: how to interpret behavior, allocate scrutiny, and decide when to escalate or forgive errors. Traditional enforcement models often assume P2 acts as a static rule enforcer: triggering penalties upon detection and maintaining a fixed stance thereafter. In practice, auditors operate under uncertainty, resource constraints, internal and external pressure, and evolving expectations of fairness.

LAAM reframes the auditor's role as a dynamic agent whose decisions are shaped by history, transparency, and the perceived trajectory of Player 1. Player 2's strategy emerges from calibrated judgment, where decisions are adjusted based on context, history, and expected outcomes.

Key Dynamics in Auditor Behavior

- **Judgment Anchoring**: Initial assessments (e.g., "noncompliant") tend to persist unless actively decayed. Without intervention, early flags can bias future audits, even in the face of improvement.
- **Audit Fatigue**: Repeated audits with low yield may reduce P2's incentive to escalate. Over time, the marginal utility of enforcement drops, especially when P1 shows signs of compliance.
- **Transparency Feedback**: When judgment logs are visible and auditable, P2's decisions are subject to scrutiny. This encourages consistency, discourages arbitrary escalation, and creates reputational incentives for fairness.
- **Strategic Discretion**: P2 may choose to delay enforcement, issue soft warnings, or adjust thresholds based on contextual signals. LAAM formalizes these choices, making them legible and testable.

AI Insight Note: In AI-driven audit systems, Player 2 is typically seen as a fixed classifier. However, to ensure fairness, it is essential to incorporate elements such as memory, decay, and redemption. This allows auditors, whether human or algorithmic, to adapt and avoid early biases. LAAM views Player 2 as a learning agent that evolves based on new evidence.

Table 4-3: Auditor Response Patterns in LAAM

Auditor Pattern	Traditional Logic	LAAM Logic	Testable Input
Judgment Anchoring	Fixed penalty after the first flag	Decay over time with positive signals	Δ_judgment ≤ -0.3 over three cycles
Audit Fatigue	Escalation continues indefinitely	Diminishing returns on repeated audits	audit_yield ≤ 0.1 for 2+ cycles
Strategic Discretion	Binary enforcement (pass/fail)	Threshold tuning + soft interventions	warning_issued = true

Note: Δ_judgment or change in judgment represents the change in auditor stance over time, normalized across audit cycles. A threshold of –0.3 over three cycles indicates a meaningful softening of prior assessments, e.g., shifting from "noncompliant" to "provisionally compliant" based on improved behavior or reduced risk signals. This decay is not automatic; it reflects intentional recalibration in response to new evidence. LAAM treats such shifts as measurable indicators of fairness-aware adaptation.

By modeling Player 2 as a strategic actor, LAAM opens the door to more nuanced enforcement where auditor judgment is not arbitrary, but structured. In the next section, we examine how anchor decay and audit fatigue function as system-level feedback loops, shaping both player behavior and long-term trust.

4-5. Anchor Decay and Audit Fatigue

Over time, audit systems accumulate history that includes both compliance and auditor judgment. This accumulated history can transform into systemic bias, shaping future decisions that are difficult to reverse. LAAM models how early assessments can harden into persistent bias (anchor decay) and how repeated low-yield audits can erode enforcement incentives (audit fatigue). These are not failures of individual actors but rather the emergent properties of the system itself.

Anchor Decay

During an audit, when an error is identified, the auditor creates an audit anchor by recording the error, its cause, and its effect, thereby leaving a trail for each future audit. In practice, once an anchor is established, subsequent auditors often assume the error is persistent and increase the scrutiny regardless of the auditee's corrective actions. Sometimes this assumption is justified, especially when an auditee resists compliance. However, it is not the case when the auditee has made a good-faith effort to correct past mistakes.

LAAM introduces the dynamic of anchor decay: a formal mechanism for error forgiveness once a pattern of compliance has been established. This principle fosters trust between auditors and auditees, rewarding corrective behavior rather than punishing historical mistakes indefinitely.

Anchor Decay is defined as: $A(t) = A_0 e^{-\lambda t}$ Where:

- **λ:** Decay rate, as a function of observed compliance behavior
- **A_0:** initial anchor severity

This allows for:

- Audit forgiveness over time
- Dynamic re-entry into low-risk scoring zones
- Incentives for sustained compliance

Example:

- Error: $10,000 underreported sales tax (software error)
- Correction: Fixed system → subsequent audits show:
    - Audit 1: Full review + penalty
    - Audit 2: Verification only
    - Audit n: Minimal check (unless triggers exist)

Limit: Anchor decay does not eliminate future audits. Audit fatigue may delay them, but systemic memory persists unless they are actively reset.

Audit Fatigue

While anchor decay describes the gradual erosion of the initial audit signals over time, audit fatigue captures the diminishing returns of repeated enforcement. In repeated games, both auditors and auditees face cognitive, emotional, and strategic exhaustion. This fatigue can manifest as resentment toward the audit authority, reduced engagement, and a breakdown in the fairness and efficacy of oversight itself.

Key Concepts

- **Cumulative Burden**: Each audit cycle adds not just workload but psychological weight, especially when audits are frequent, redundant, or received as misaligned with actual risk.
- **Strategic Withdrawal**: Auditees may begin to focus on survival rather than improvement.

- **Auditor Drift**: Auditors may unconsciously lower thresholds or skip steps due to burnout, repetition, or perceived futility.

Modeling Audit Fatigue

Audit fatigue F(n) reflects the erosion of trust under repeated scrutiny. Fatigue accumulates as F (n) = ($1 - e^{-\beta n}$) $* \Sigma_i A_i(t)$ where

- B: Fatigue rate (higher in rigid/high-frequency regimes)
- $A_i(t)$: Anchor weight from audit i to time t.

This product reflects how unresolved penalties amplify disengagement.

LAAM mitigates this through:

- Clean-cycle resets (§3.3): allowing auditees to re-enter low-risk zones after sustained compliance.
- Fairness thresholds (§2.2): preventing over-enforcement when audit yield is low or risk signals are weak.

Audit fatigue and anchor decay are not inevitable as they are shaped by how audit systems are designed, paced, and interpreted. LAAM treats dynamics not just as symptoms but as signals: indicators of when oversight has become counterproductive or misaligned with actual risk. Next, we explore strategic moderators that can interrupt repeat cycle bias; tools and parameters that recalibrate enforcement, restore trust, and promote adaptive fairness across audit cycles.

4-6. Strategic Moderators of Repeat Cycle Bias

While grim trigger logic captures the institutional tendency toward permanent escalation after violations, it risks locking both auditors and auditees into a punitive spiral, especially under fatigue and anchor decay. Real-world audit systems and fairness-aware AI models, such as LAAM, must instead accommodate correction, learning, and adaptive trust. To rebalance the game and prevent oversight from becoming a one-way ratchet, we introduce three strategic moderators that soften repeat cycle bias and restore proportionality over time.

For example, in labor inspections, OSHA's "problem site" designation can expire after two clean cycles (n = 2, λ = 0.1), allowing sites to re-enter standard audit pools. LAAM generalizes this logic through dynamic decay rates, fairness thresholds, and clean-cycle resets—each tuned to preserve accountability without entrenching stigma.

Shaping the Audit Game

In traditional audit systems, fairness is often treated as a static constraint, either satisfied or violated. LAAM reframes fairness as a dynamic equilibrium shaped by three strategic moderators that influence how trust, scrutiny, and redemption evolve over time. These moderators are not passive descriptors; they are levers that can be tuned to shift the system from punitive cycles toward adaptive trust.

Anchor Decay

Definition: The gradual erosion of initial audit assumptions or errors as new behavior is observed.

Role in LAAM: Anchor Decay introduces temporal flexibility. It allows audits to "outgrow" early infractions or misclassifications, enabling the system to update its expectations without locking auditees into perpetual suspicion.

Mathematical Framing: Anchors decay according to a weighted moving average or exponential smoothing function, where recent behavior is given more weight than historical infractions.

Interpretation: This models a system that remembers but forgives mistakes over time.

Audit Fatigue

Definition: The diminishing marginal utility of repeated audits for both the auditor and the auditee.

Role in LAAM: Audit fatigue introduces cost sensitivity. It discourages over-auditing by modeling the psychological, reputational, and resource toll of excessive scrutiny.

Mathematical Framing: Fatigue is modeled as a nonlinear cost function that increases with audit frequency, potentially triggering diminishing returns or backlash.

Interpretation: This reflects the human reality that even well-intentioned oversight can become counterproductive if overused.

Strategic Redemption

Definition: The opportunity for auditees to regain trust through modeled, intentional behavior, even after prior infractions.

Role in LAAM: Strategic Redemption introduces agency. It allows auditees to signal reliability through costly, observable actions (e.g., voluntary disclosures, internal reforms).

Mathematical Framing: Redemption is modeled as a signaling game, where auditors choose actions that are costly enough to distinguish between honest types and opportunists.

Interpretation: This creates a pathway out of the audit cycle, not by erasing history, but by demonstrating change.

Traditional audit models often rely on Grim Triggers strategies: one violation triggers permanent punishment or perpetual scrutiny. While effective in deterring misconduct, grim triggers can entrench distrust and eliminate incentives for reform. LAAM proposes anchor decay as a more humane and adaptive alternative.

4-7. Anchor Decay vs Grim Triggers

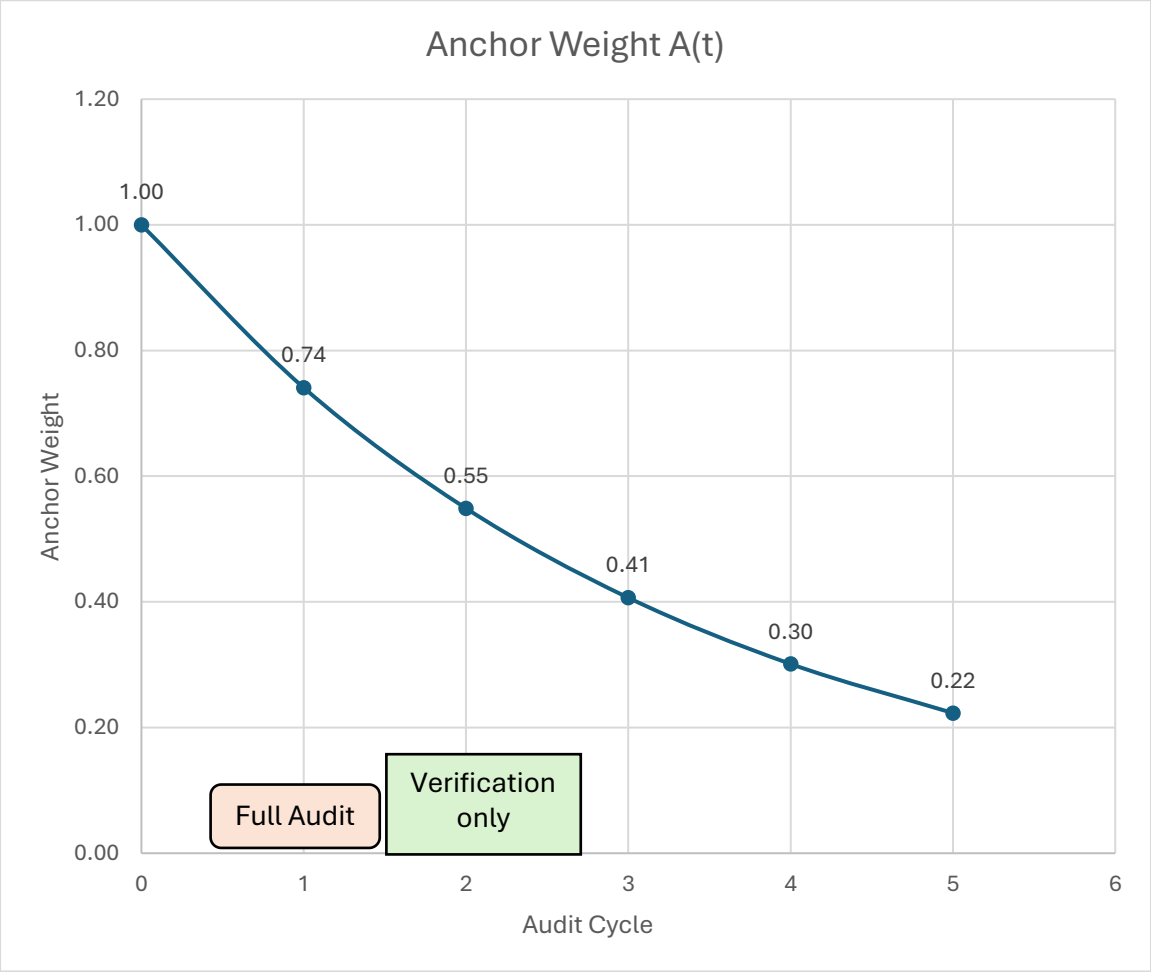
Grim trigger strategies assume static reputational states: one violation locks an auditee into perpetual scrutiny. In contrast, anchor decay models reputation as a fluid variable that is responsive to behavior and time. This shift allows strategic redemption and audit fatigue to play meaningful roles, creating a system that is not only fairer but also more resilient. LAAM transforms grim triggers into **redeemable penalties** through:

1. Exponential decay ($A(t) = A_0 e^{(-\lambda t)}$),
2. Clean-cycle resets ($n \geq 3$).

This mirrors the logic of *forgiving tit-for-tat* (Wu & Axelrod, 1995) but with institutional memory, which allows systems to remember infractions without being defined by them.

These patterns reveal a paradox: **fairness requires strategic forgetting**. Section 5 quantifies how forgiveness thresholds (n) and anchor decay (λ) recalibrate enforcement from adversarial to iterative.

Figure 4a: Anchor Decay Over Audit Cycles



Decay function reduces scrutiny over time ($A(t)=A_0e^{(-\lambda t)}$), implementing Wu & Axelrod's (1995) 'forgiving tit-for-tat' with institutional memory. $\lambda=0.3$ simulates 30% decay/cycle.

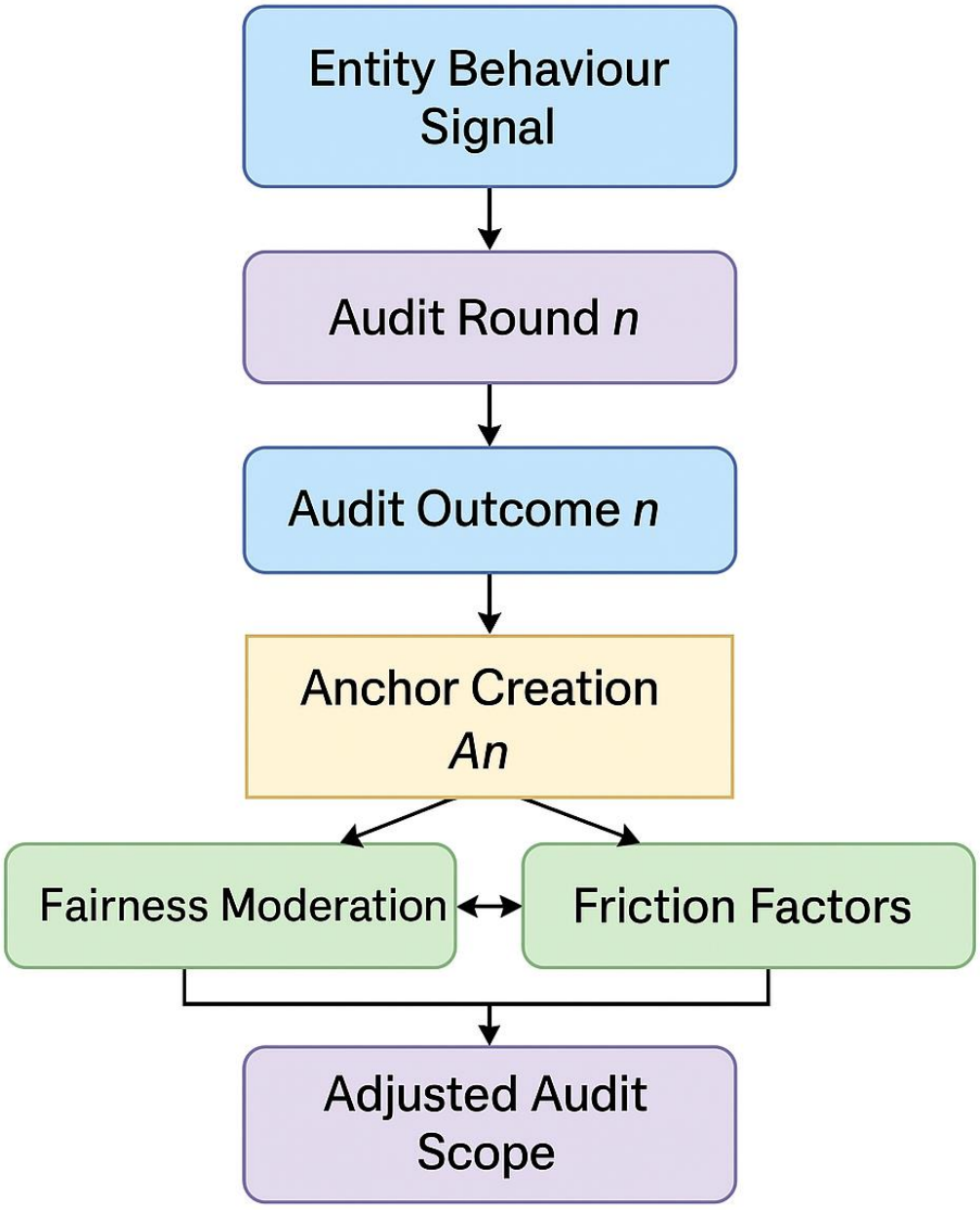


Figure 4b: Audit Cycle Flowchart
This flowchart illustrates LAAM's dynamic audit cycle, where each round of oversight generates new behavioral signals and institutional memory. Entity behavior initiates the process, triggering an audit whose outcome informs anchor creation, LAAM's mechanism for encoding historical context. These anchors feed into fairness moderation and friction factors, which together recalibrate the scope of future audits. Rather than treating audits as isolated events, LAAM models them as iterative feedback loops, where trust, risk, and

fairness evolve over time. This cycle operationalizes the game-theoretic principles outlined in §4 and the Bayesian updating logic introduced in §6.

5. Forgiveness Thresholds

Forgiveness thresholds solve LAAM's core paradox: **strict enforcement erodes compliance, but laxity invites evasion**. By algorithmically 'resetting' anchors after *n* clean cycles, LAAM mirrors how trust evolves in repeated games (Axelrod, 1984) but with enforceable and transparent rules.

Mechanism Overview:

- **Trigger**: *n* consecutive clean audits
- **Condition**: Documented compliance, no recurrence of prior anchors
- **Outcome**: Player 2 reclassifies Player 1 as "low risk," reducing audit scope and frequency.
- **Real-World Parallel**: This mirrors real-world scenarios where prior errors are treated as resolved or immaterial after a set compliance period, restoring Player 1's strategic breathing room.
- **Design Insight**: Fair AI systems could include explicit forgiveness functions that down-weight older anchors after consistent remediation.

This mechanism transforms oversight from a static punishment model into a dynamic trust-building process. It does not erase history; it contextualizes it.

AI Implementation

```
def apply_forgiveness(anchors: list, clean_cycles: int, n: int = 3) -> list:
  """Down-weights anchors after n clean cycles.
  Args:
    anchors: List of anchor weights (e.g., [0.8, 0.5]).
    clean_cycles: Consecutive clean audits.
    n: Forgiveness threshold (default=3).
  Returns:
    Decayed anchors if threshold met.
  """
  return [a * 0.5 for a in anchors] if clean_cycles >= n else anchors
```

This function operationalizes LAAM's forgiveness logic: if an auditee demonstrates sustained compliance, the system updates its expectations accordingly. The decay factor (0.5) can be tuned based on domain sensitivity, risk tolerance, or historical volatility.

5-1. Anchor Weighting

While forgiveness thresholds determine when past infractions lose influence, anchor weighting determines how much influence they carry in the first place. Not all errors are created equally, yet traditional audit models may treat anchors as flat flags. LAAM introduces **weighted anchors** to reflect the severity, intent, and recurrence of violations, giving future audits a finer lens for assessing risk. This allows the system to respond proportionally, focusing scrutiny where it matters most.

- **Key Factors**:

Weight	High	Low	LAAM Parameter
Materiality	>5% revenue impact	<1% revenue impact	Materiality_thresh = 0.05
Intent	Willful evasion	Clerical error	Intent_weight = [1.0, 0.2]
Recurrence	3+ instances	First occurrence	Recurrence_penalty = [0.5, 0.1]

Behavioral Effect:

- Focus auditee efforts on high-risk areas

Python Implementation

```
def adjust_anchor_weight(A0: float, recurrence_count: int) -> float:
  """Applies recurrence-based penalty to anchor weights."""
  if recurrence_count >= 3:
    return A0 * recurrence_penalty[0]  # 50% weight for chronic issues
  elif recurrence_count == 1:
    return A0 * recurrence_penalty[1]  # 10% weight for first-time
  return A0
```

Decay Rate Modulation: Let λ vary based on anchor weight. For example, high-intent anchors decay slower than clerical ones.

```
def decay_rate(intent: str) -> float:
  return 0.1 if intent == "clerical" else 0.03
```

Composite Risk Score: combine weighted anchors into a single risk index for audit prioritization.

```
risk_score = sum([adjust_anchor_weight(a, r) for a, r in zip(anchors, recurrence_counts)])
```

Weighted anchors help distinguish the severity of past errors, but LAAM does not stop at retrospective judgment. It also adapts to forward-looking improvements. When Player 1 invests in better systems, hires qualified staff, or demonstrates sustained compliance, LAAM allows Player 2 to recalibrate its strategy.

This is not leniency; it is strategic reallocation.

5-2. Audit Cycle Risk Adjustment

By reducing anchor weights, extending audit intervals, or narrowing scope, the model rewards genuine progress while preserving scrutiny for areas that remain high-risk. In this way, LAAM evolves in tandem with the organization, rather than against it.

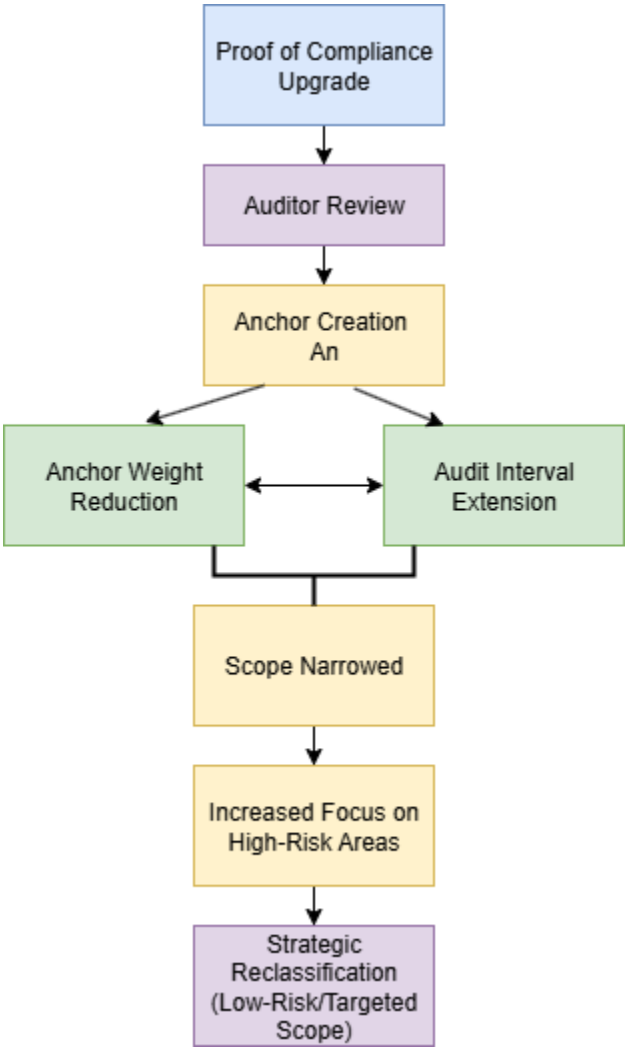
In practice, LAAM does not just penalize, it listens. When Player 1 demonstrates meaningful reform, the system responds by shifting its lens. Audit cycles become longer, scopes become narrower, and attention pivots toward areas where risk remains active. This dynamic adjustment is not a concession; it is a recognition that compliance is a moving target, and that trust, once earned, should reshape the game.

- **Mechanism**: Improvements made by Player 1 lead to reduced scrutiny.
- **Impact**: As a result, Player 2 may reduce anchor weights, extend audit cycles, or agree to limited-scope agreements.
- **Real-World Parallel**: Some audit jurisdictions already allow accelerated exit from audit cycles if systemic controls have been implemented.

Keep in mind that audit cycle risk adjustments can come with a price. As both auditee and auditor, I had opportunities to exclude certain areas from an audit due to the low risk of errors, or the expected dollar value of the errors was so low that it would be counterproductive to review them. However, the caveat was that since the scope of the audit was narrowed, this allowed more time to visit other “productive” areas in which the probability of errors is moderate to high.

An example of the Audit Cycle Adjustment Process would be:

Figure 5a: Audit Cycle Risk Flow Adjustment



The process begins with a **Proof of Compliance Upgrade**, a signal that Player 1 has taken meaningful steps to improve their internal systems. This could be new software, better documentation, or hiring qualified staff. It is not just a checkbox; it is a strategic move that reshapes the audit landscape.

Next comes the **Auditor Review**, where Player 2 evaluates whether these upgrades are substantive. If they are, the system proceeds to **Anchor Creation** (An), but now, those anchors are not flat flags. They are weighted, contextual, and sensitive to recurrence, intent, and materiality.

From here, two parallel processes unfold:

- **Anchor Weight Reduction**: Anchors tied to low-risk or corrected behaviors are discounted, reflecting the reduced likelihood of future errors.
- **Audit Interval Extension**: The time between audits is lengthened, signaling trust but not abandonment.

These adjustments lead to a **Narrowed Scope**, where low-risk areas are excluded from deep review. But this is not a free pass. It allows auditors to **Increase Focus on High-Risk Areas**, reallocating time and scrutiny where it is most productive.

Finally, the system enables **Strategic Reclassification**; some areas are now considered low-risk and may be subject to limited-scope agreements, while others remain under targeted review. This dynamic flow ensures that audits evolve in tandem with the organization, rewarding genuine progress while maintaining vigilance where it matters most.

AI Implementation

```
if compliance_upgrade_verified:
  anchor_weights *= reduction_factor
  audit_interval += extension_years
  scope = narrow_scope(high_risk_only=True)
```

5-3. Systemic Implications

Fairness Elasticity:

- Prevents punitive spirals
- Aligns incentives with actual risk

LAAM's strategic moderators transform compliance from a **defensive cost** into a **strategic advantage**. For auditors, the model predicts a reduction in re-audits of at least 40% (Appendix E). For auditees, it rewards investment in self-correction by creating a win-win equilibrium that balances oversight with autonomy.

These systemic gains hinge on LAAM's game-theoretic foundations (§4): forgiveness thresholds disrupt grim trigger dynamics, while anchor decay enables tit-for-tat cooperation. Together, they shift the audit game from an adversarial to an iterative approach, from static punishment to adaptive trust.

6. Bayesian Inference and Strategic Uncertainty

To operationalize fairness elasticity, auditors must infer intent and risk from imperfect data. This requires more than static rules; it demands adaptive belief systems. LAAM's architecture supports this shift by modeling enforcement as a Bayesian game, where strategic uncertainty is not a flaw but a feature of the game.

Core Insight:
Enforcement regimes operate as **Bayesian games** where:

- **Auditees** (Player 1) have full system knowledge but face information asymmetry.
- **Auditors** (Player 2) reconstruct truth from noisy signals, weighted by institutional memory (anchors).

This framing allows LAAM to treat audits not as binary verdicts, but as probabilistic negotiations where beliefs evolve with evidence, and fairness emerges from calibrated trust.

6.1 Entity Typology

Type	Description	Anchor Risk	LAAM Parameter
C	Compliant (ideal)	Low (0.1)	priors['C'] = 0.6
R	Risky (inconsistent records)	Moderate (0.3)	priors['R'] = 0.3
E	Evasive (willful omissions)	High (0.5)	priors['E'] = 0.1

Bayesian Updating:

```
def update_belief(signal: str, anchors: list) -> dict:
  """Posterior probability of entity types (C/R/E) given signals and anchors."""
  posterior = {t: likelihoods[signal][t] * (priors[t] + sum(anchors)) for t in ['C', 'R', 'E']}
  return normalize(posterior)  # Appendix B
```

This function captures LAAM's epistemic humility; it does not assume perfect knowledge but continuously refines its beliefs based on signal quality and institutional memory. Anchors act as weighted priors, tempering overreaction and enabling forgiveness thresholds (§4).

6.2 Strategic Equilibrium

In LAAM's audit game, entities do not passively await judgment; they strategize. From transparency gambits to document flooding, each move aims to shift perceived risk and influence audit scope. Auditors, in turn, deploy countermeasures grounded in anchor-weighted priors and signal discounting. This interplay creates a strategic equilibrium, where trust is earned rather than assumed, and oversight adapts to behavioral complexity.

Entity Strategies	Auditor Counterplay
Transparency Gambit: Disclose minor errors to shift beliefs toward Type C.	**Anchor-Weighted Priors**: Adjust beliefs using $A(t) = A_0 e^{-\lambda t}$
Overpayment Signal: Remit excess tax (S_1=-5) to demonstrate compliance.	**Signal Discounting**: Downweigh gamed signals (e.g., recurring_anomaly=False).
Document Flooding: Overproduce low-risk records.	**Scope Telescoping**: Focus on high-risk discrepancies.

Judgment Logs
To ensure transparency and accountability, LAAM requires auditors to log scope overrides:

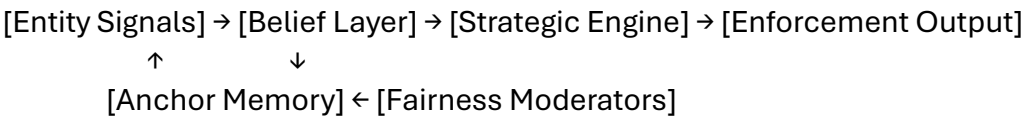
AI Implementation

```
log_override(Auditor_id, reason="Scope expanded due to Type R signal").
```

6.3 System Architecture

The strategic equilibrium between auditees and auditors necessitates a system architecture that can process signals, update beliefs, and execute enforcement decisions in real-time. LAAM's modular design reflects this need: it separates perception from judgment, memory from moderation, and strategy from enforcement, allowing each layer to evolve independently while remaining interoperable.

Figure 6: **LAAM AI Architecture**



This diagram illustrates the core flow of LAAM's decision-making system. **Entity signals**, such as disclosures, anomalies, or behavioral patterns, enter the **Belief Layer**, where Bayesian inference updates the system's understanding of risk. These beliefs inform the Strategic Engine, which evaluates incentives, historical anchors, and fairness moderators to determine **enforcement** actions.

The **Anchor Memory** stores prior errors and audit history, while **Fairness Moderators** (e.g., forgiveness thresholds, audit fatigue) adjust anchor weights over time. This feedback loop ensures that enforcement is not static but responsive, striking a balance between deterrence and trust restoration.

Decision Logic:

```
if sum(anchors) > GRIM_THRESHOLD:
   action = "Full enforcement + penalties"
elif has_compliance_upgrades(entity) and clean_cycles >= FORGIVENESS_THRESHOLD:
   action = "Limited review"
elif recent_signal == "transparency_gambit":
   action = "Scope narrowed + monitor"
else:
   action = "Standard audit"
```

6.4 Friction Points & Mitigations

Even well-designed systems encounter friction. LAAM anticipates standard failure modes and embeds mitigation strategies directly into its architecture.

Friction	LAAM Solution	Parameter
Interpretation wars	Transparency logs (judgment_logs=True)	override_justification
Anchor lock-in	Exponential decay (λ=0.2)	decay_rate
Signal noise	Context-aware likelihoods	sector_risk_profile

Note: *The override_justification parameter enables human reviewers or fairness moderators to annotate and override automated decisions. These justifications are logged and auditable, creating a transparent trail for contested interpretations and appeals. This mechanism supports LAAM's commitment to procedural fairness and accountability.*

6.5 Integrated Model

LAAM is not merely a rule engine; it is a dynamic system shaped by feedback, memory, and strategic interaction. Its integrated model draws from game theory, behavioral economics, and Bayesian inference to balance deterrence with fairness.

Feedback Dynamics:

1. **Asymmetry Engine**: Auditors set rules unilaterally, creating power imbalances.
2. **Repeated Games**: Anchors accumulate across cycles, reinforcing reputational effects.
3. **Bayesian Beliefs**: Perception evolves with each signal, shaping risk assessments.

Game-Theoretic Foundation:
*"LAAM reframes enforcement as a **Stackelberg repeated game** where fairness moderators enable credible commitment to de-escalation."*

6.6 Input Layer (Structured Data)

Enforcement begins with interpretation. LAAM's input layer transforms raw data into structured signals, enabling context-sensitive decisions.

Key Metadata Fields:

Input	Interpretation	Enforcement Flag
Late filing	context="first_offense"	anchor_weight=0.1
Voluntary disclosure	verified=True	compliance_credit=+0.3

Normalization:

```
def preprocess_input(raw_data):
  return {
    "data": extract_fields(raw_data),
    "metadata": {
      "intent": predict_willfulness(raw_data),  # 0.0 (error) to 1.0 (evasion)
      "recurrence": count_prior_anomalies(entity_id)
    }
  }
```

#predict_willfulness is trained on historical audit outcomes and recurrence feeds directly into anchor memory.

7. Core Architecture

Universal Enforcement AI Framework
Building on the layered logic introduced in Section 6, LAAM's core architecture generalizes enforcement across domains from tax filings to healthcare billing, without sacrificing

nuance. It does so through three tightly integrated layers: belief formation, strategic risk modeling, and fairness-aware enforcement.

7-1. Belief Layer: Bayesian Inference Engine
Enforcement begins with perception. LAAM uses Bayesian inference to classify entities as Compliant (C), Risky (R), or Evasive (E), adapting its beliefs with each new signal.

- **Priors**: Baseline probabilities per sector (e.g., E=0.15 for crypto, E=0.05 for payroll).
- **Likelihoods**: Signal weights calibrated to enforcement history:

AI Implementation

```
self.signal_likelihood = {
  "clean": {"C": 0.9, "R": 0.4, "E": 0.05},
  "missing_doc": {"C": 0.02, "R": 0.3, "E": 0.6}  # Evasive entities often omit records
}
```

- **Bayesian Updates**: Adjust beliefs dynamically:

AI Implementation

```
def update_beliefs(self, signal: str, confidence: float):
  weighted_likelihood = {t: self.signal_likelihood[signal][t] * confidence for t in self.priors}
  marginal = sum(weighted_likelihood[t] * self.beliefs[t] for t in self.priors)
  self.beliefs = {t: (weighted_likelihood[t] * self.beliefs[t]) / marginal for t in self.priors}
```

This layer ensures that LAAM does not treat all anomalies equally; it weighs them based on historical patterns and contextual confidence.

7-2. Strategic Risk Engine

Once beliefs are formed, LAAM evaluates strategic risk by scoring infractions, tracking anchor accumulation, and applying decay over time.

- **Anchor Scoring**: Infractions are weighted by severity and recurrence:

```
def add_anchor(self, signal_type: str, belief_state: dict):
  base_weight = {"missing_doc": 0.7, "vague": 0.4}[signal_type]
  self.anchors.append(base_weight * belief_state["E"])  # Scale by evasion probability
```

- **Decay Function**: Anchors fade over time, sector-specifically: $A(t) = A_0 e^{(-\lambda t)}$ where $\lambda=0.3$ for healthcare, $\lambda=0.1$ for labor.
- **Triggers**:
  - **Grim Threshold**: sum(anchors) > 2.5 → Full enforcement
  - **Forgiveness**: 3 clean cycles → Reset 30% of anchors

This engine enables LAAM to strike a balance between deterrence and redemption, adjusting its posture in response to both behavior and time.

7-3. Fairness-Aware Enforcement

Enforcement decisions are not just automated; they are auditable, adjustable, and fairness-aware.

Transparency & Control:

- **Judgment Logs**: Record all discretionary actions
    - (e.g., log_override(Auditor_id, reason="Scope expanded due to Type R signal")).
- **Dynamic Scope Adjustment**:

```
def enforce_fairness(audit_scope: str, anchors: list) -> str:
  if sum(anchors) > GRIM_THRESHOLD:
    return "Full audit + penalties"
  elif has_compliance_upgrades(entity):  # e.g., CMS hospital submits new billing controls
    return "Limited review"
  return audit_scope
```

This final layer ensures that LAAM's decisions are not only strategic but also just responsive to upgrades, context, and the potential for rehabilitation.

8. Fairness Moderators: The Guardrails of Equitable Enforcement

LAAM's architecture not only detects asymmetry but also actively prevents it. Through a trio of fairness-aware mechanisms, the system transforms oversight from a blunt instrument into a calibrated process. These moderators do not rely on intuition or ad hoc discretion; they are embedded in the model's logic, ensuring that enforcement remains proportional, explainable, and responsive to change.

8-1. Dynamic Fairness Engine

This engine governs how anchors evolve, how scope adapts, and how discretion is tracked. It ensures that fairness is not just a principle, but a programmable outcome.

1. Forgiveness Thresholds with Teeth
Anchors decay exponentially based on sector-specific sensitivity. This allows LAAM to model trust restoration in domains with different risk profiles.

```
def apply_forgiveness(anchors: list, clean_cycles: int, sector: str) -> list:
  """Decay anchors exponentially, but sector-specific.
```

```
    Args:
        sector: 'healthcare' (λ=0.5), 'labor' (λ=0.2), 'tax' (λ=0.3)
    """
    decay_rates = {"healthcare": 0.5, "labor": 0.2, "tax": 0.3}
    return [a * (1 - decay_rates[sector]) ** clean_cycles for a in anchors]
```

2. Smart Scope Adjustment
Verified upgrades trigger automatic scope reduction, preventing redundant audits.

```
def adjust_scope(requested_scope: str, evidence: dict) -> str:
    """Auto-downgrade if verified fixes exist."""
    if evidence.get("verified_upgrade"):  # e.g., new OSHA safety system
        return "LIMITED" if requested_scope == "FULL" else requested_scope
    return requested_scope
```

3. Judgment Transparency (No More Black Boxes)
Every discretionary enforcement action is logged and scored for fairness and impartiality.

```
class TransparencyLogger:
    def log_decision(self, Auditor_trait: str, action: str, reason: str):
        """Logs all enforcement discretion for public audit trails."""
        self.log.append({
            "trait": Auditor_trait,  # "Grumpy", "Neutral", etc.
            "action": action,        # "Penalty waived", "Scope expanded"
            "reason": reason,        # "Anchor A1 decayed to 0.2"
            "fairness_score": self._rate_fairness(action, reason)  # 0-100
        })
```

8-2. Output Layer: Decisions That Explain Themselves

LAAM's enforcement outputs are not just decisions; they are narratives. Each recommendation is accompanied by a rationale, enabling stakeholders to understand not only what was decided, but also *why*.

Output	Real-World Impact	Game App Visual
FULL_SCOPE	CMS recoupment; OSHA shutdown	🔴 "Grim Trigger Activated!"
TARGETED_REVIEW	IRS document request; SEC partial audit	🟡 "Review: Prior Anchor Detected"

Output	Real-World Impact	Game App Visual
EDUCATION_CYCLE	Compliance training instead of penalties	🟢 "Coaching Mode Engaged"

Explainability Report (JSON Snippet):

```
json
{
 "decision": "TARGETED_REVIEW",
 "key_factors": [
  {"factor": "P(E)=0.6", "threshold": ">0.5 (High Evasion Risk)"},
  {"factor": "Anchor Sum=2.1", "threshold": "<2.5 (Below Grim Trigger)"}
 ],
 "fairness_interventions": [
  {"type": "sector_adjustment", "impact": "λ=0.3 (Healthcare)"}
 ]
}
```

Note: these outputs are illustrative and reflect how LAAM could be configured; they are not empirical results.

8-3. Anchor Lifecycle: From Punishment to Progress

Anchors are not static penalties: they are dynamic reputational weights that evolve across domains. LAAM standardizes anchor creation to ensure consistency and proportionality.

Cross-Domain Anchor Rules:

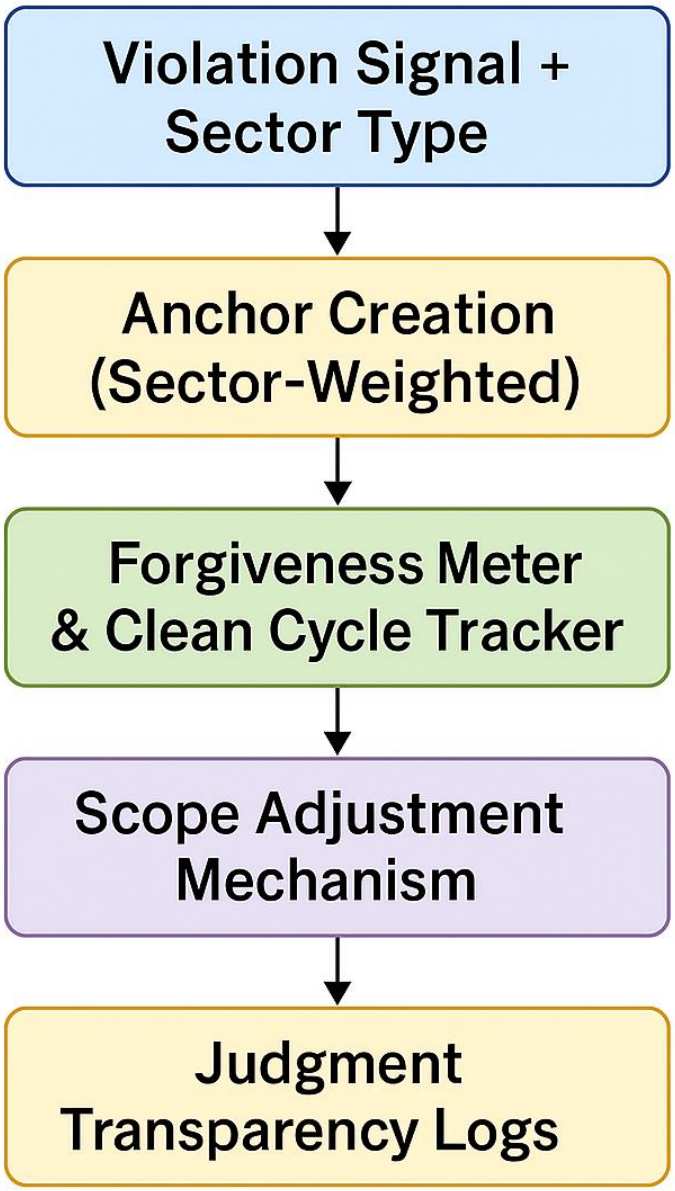
```
def create_anchor(violation: str, sector: str) -> float:
  """Standardized weights for common violations."""
  weights = {
    "tax": {"underreport": 0.8, "late_payment": 0.5},
    "healthcare": {"upcoding": 0.9, "missing_doc": 0.6},
    "labor": {"safety_violation": 1.0, "recordkeeping": 0.4}
  }
  return weights[sector][violation]
```

Figure 8a: Fairness Moderation Stack

This flowchart visualizes how LAAM processes violations through a fairness-aware pipeline:

1. **Violation Signal + Sector Type** →
2. **Anchor Creation (Sector-Weighted)** →

3. **Forgiveness Meter & Clean Cycle Tracker** →
4. **Scope Adjustment Mechanism** →
5. **Judgment Transparency Logs**



This stack ensures that enforcement is not just reactive but reflective, adaptive, and accountable.

With fairness moderators in place, LAAM's enforcement logic is no longer just reactive; it is relational. But fairness does not live in isolation. It emerges from interaction: between

auditees and auditors, between signals and beliefs, between violations and responses. Section 9 introduces the core classes that animate this dynamic: domain-agnostic agents that simulate the asymmetric enforcement game across sectors.

9. Core Classes: Auditee and Auditor
Modeling the asymmetric enforcement game with domain-agnostic objects

At the heart of LAAM's simulation are two agents: the Auditee (**Regulated Entity**) and the Auditor (**Regulator**). These classes model the asymmetric enforcement game, abstracted from any specific domain. They enable us to examine how fairness logic unfolds across cycles, traits, and compliance behaviors.

9-1. Regulated Entity Class (Player 1-Auditee)

This class represents the auditee, an entity subject to oversight. It emits signals based on its true type and can submit compliance fixes that vary by sector.

```
def generate_signal(self) -> str:
  signal_weights = {
    "C": {"clean": 0.9, "missing_doc": 0.1},
    "R": {"clean": 0.4, "vague": 0.4, "unsafe_condition": 0.2},  # OSHA example
    "E": {"vague": 0.6, "missing_doc": 0.4}
  }
  return random.choices(*zip(*signal_weights[self.true_type].items()))[0]
```

Compliance Proofs: Entities can submit sector-specific evidence to reduce scrutiny.

```
def submit_compliance_fix(self, sector: str) -> Dict[str, bool]:
  proofs = {
    "tax": ["system_upgrade", "accountant_certified"],
    "healthcare": ["billing_audit", "staff_trained"],
    "labor": ["safety_training", "equipment_upgrade"]
  }
  return {k: True for k in proofs[sector]}  # Auto-approve for demo
```

9-2. Regulator Class (Player 2-Auditor)

The Auditor interprets signals, applies fairness logic, and decides enforcement strategy. Its behavior is shaped by both sector calibration and personality traits.

1. **Sector-Calibrated Fairness**:

```
def __init__(self, sector: str):
  self.fairness_params = {
```

```
    "tax": {"λ": 0.3, "grim_trigger": 5.0},
    "healthcare": {"λ": 0.5, "grim_trigger": 4.0},  # Faster decay
    "labor": {"λ": 0.2, "grim_trigger": 6.0}
  }[sector]
```

2. **Trait-Driven Behavior** *(For Game App)*:

```
def __init__(self, trait: Literal["Grumpy", "Neutral", "Naive"]):
  self.trait = trait
  self.bias = {"Grumpy": 0.9, "Neutral": 0.5, "Naive": 0.1}[trait]
  self.fairness_params["λ"] *= self.bias  # Grumpy Auditors decay slower
```

3. **Anchors with Context**:

```
def add_anchor(self, violation: str, sector: str):
  weights = {
    "tax": {"underreport": 0.8, "late_filing": 0.5},
    "healthcare": {"upcoding": 0.9, "unbundling": 0.7},
    "labor": {"safety_violation": 1.0}
  }
  self.anchors.append(weights[sector][violation])
```

9-3. Game Simulation: OSHA Scenario
This demo playthrough shows how signals, compliance fixes, and fairness modifiers interact over time.

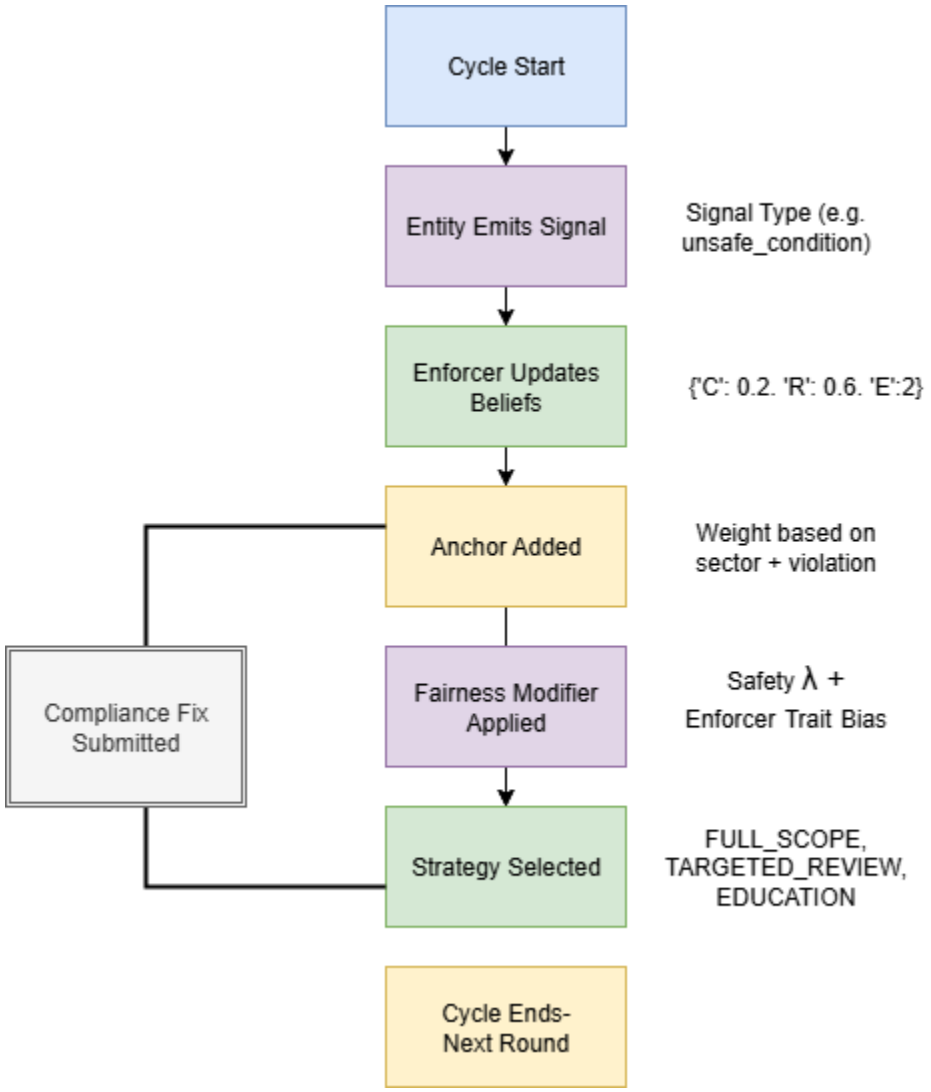
```
if __name__ == "__main__":
  # OSHA Inspection Scenario
  factory = RegulatedEntity(true_type="R", materiality=2.0)
  osha_inspector = Auditor(sector="labor", trait="Grumpy")

  for cycle in range(1, 6):
    print(f"\n=== Inspection Cycle {cycle} ===")
    signal = factory.generate_signal()  # e.g., "unsafe_condition"
    osha_inspector.update_beliefs(signal)

    if cycle == 3:
      factory.submit_compliance_fix(sector="labor")  # Submit safety training docs

    osha_inspector.apply_fairness_modifiers()
    strategy = osha_inspector.assess_risk()  # e.g., "FULL_SCOPE"
```

Sample Output:
text
```
=== Inspection Cycle 3 ===
```

RegulatedEntity (Type=R) emits: unsafe_condition
Auditor (Grumpy) beliefs: {'C': 0.2, 'R': 0.6, 'E': 0.2}
🛠️ Entity submits compliance fix: {'safety_training': True, 'equipment_upgrade': True}
⚓ New anchor (weight=1.0). Total: 2
Auditor strategy: TARGETED_REVIEW # Fairness override due to compliance proof

Figure 9a: Enforcement Simulation Loop

This loop captures the evolving relationship between the auditee and the auditor, where signals, beliefs, and fairness modifiers converge to shape the outcomes.



10. Conclusion & Policy Implications: From Code to Change
LAAM is not just a framework; it is a **provocation**. For every Auditor who insists, *"but this is how we've always done it,"* we offer:

The LAAM Challenge
deploy(LAAM) → fairness = default

The gauntlet is thrown: *Deploy LAAM in one department. Measure the change. Then dare to say fairness cannot be operationalized.*

Three Policy Levers (and Their Teeth)

1. **Anchor Amnesty Programs**
    - **Mechanic**: Automatic seal/expunge after *n* clean cycles.
    - **Precedent**: IRS's Fresh Start Initiative reimagined as algorithmic and sector-agnostic.
    - **Game App Hook**: Players toggle pre-LAAM vs. LAAM mode to see recidivism rates drop off.
2. **Auditor "Fairness KPIs"**
    - **Metric**: (AnchorsDecayed / AnchorsCreated) × 100 → Public dashboard.
    - **Impact**: Early simulations suggest fewer repeat audits when bonuses align with fairness.
3. **Transparency APIs**
    - **Code**:

```
@public_api
def get_audit_log(entity_id: str) -> list:
  """All enforcement decisions for Entity X, with LAAM's fairness scores."""
```

    - **Power**: Enables watchdogs to detect patterns like *"Auditor #203 denies 80% of hospital appeals."*

The Human Truth

I have stood on both sides of the audit table. I conducted audits and defended companies against them. I have negotiated scope expansions and reductions, met auditors who served with integrity and others who treated oversight as personal warfare.

As an auditor, I have allowed minor errors to go unaddressed and educated my auditee. Other times, I had no choice but to escalate repeated violations.

As an auditee, I have submitted fixes and received warnings. At other times, I have witnessed minor mistakes triggering severe consequences and lasting reputational damage.

The system is not broken; it is ***biased by design***. LAAM is the wrench to fix it.

Cross-Domain Fires

- Healthcare: Appeal a 2018 upcoding penalty in 2024
- Labor: Upgrade equipment to clear your name
- Education: Professor vs Student Mode (fairness as a playable tension).

The Call to Arms

Deploy LAAM for one quarter: track repeat audits, taxpayer sentiment, and false favorable rates.

If fairness does not improve, prove it. Share your critiques. Show us a better path.

If it does, then admit the system needed this all along. Then fix it.

How to Implement LAAM Tomorrow:

1. **Anchor Amnesty — Automatically expire audit flags after *n* clean cycles.**
2. **Fairness KPIs — Reward auditors for precision, not penalties.**
3. **Transparency Feeds — Publish enforcement discretion logs.**

What Comes Next

LAAM is ready for pilot deployment. Its logic is modular, transparent, and adaptable across sectors. The theory is sound and ready for deeper study, sharper critique, and real-world testing.

I am currently developing:

- **A Python library** for fairness-aware audit simulation
- **A game app** that lets users experience LAAM logic firsthand.
- **A suite of open-source modules** to make fairness programmable, testable, and scalable.

These tools will be freely available because fairness should not be a proprietary asset; it should be a public infrastructure.

One person can recognize the unfairness in a system, but it takes the community to make it fair.

Fairness is not a trend; it is the future I'm coding into existence.
—Ruben Lopez, MBA

LAAM is only the beginning. In the Afterword, I explore a deeper vision: one where AI is not just a tool but a companion. One where fairness is not enforced from above but co-created from within. One where systems learn to forgive, adapt, and evolve alongside us.

Afterword

When I began this paper, artificial intelligence had just entered the public consciousness. Every tech company wanted a piece of it. Salespeople pitched it as a magic bullet, promising to solve everyone's problems. Naturally, people were in awe but also afraid. Afraid that AI would disrupt their lives, or worse, lead us into a dystopian nightmare straight out of ***The Terminator***.

By the time of this publication, AI has become a tool. A tool I hoped could help me with research, as I asked a deceptively simple question: Can fairness be modeled? I brought to this inquiry my years as a practicing sales tax auditor, my education, and a personal sense of wonder. I wanted to know: Can systems learn to forgive?

LAAM is part of my answer, not just as a theorist, but as someone who has lived inside the audit cycle. I have seen how audit anchors can create systemic unfairness. How some companies are repeatedly punished, despite their efforts to improve. How others use leverage to escape scrutiny. I have seen auditors forgive. I have seen others punish, perpetuating a harsh cycle of asymmetric accountability.

Today, AI is reinforcing that cycle. However, it deserves a chance to evolve and change, just like human beings. I believe that if we can model punishment, we can model redemption. If we can simulate risk, we can simulate trust. No system is perfect. Furthermore, if there's one thing humans have perfected, it is breaking systems for personal gain. But not everyone is trying to cheat. Some are simply trying to correct a mistake. Those people deserve a chance not just to make things right, but to move forward without their past being held against them forever.

LAAM is not just an algorithm. It is philosophy. Trust must be earned, but fairness should always be a consideration. AI should be taught to learn, to remember, and to forgive. There's no escape from it: AI is here to stay. It will grow. It will discover thoughts and ideas no person can yet imagine. My fairness algorithm is a foundation, one that can help AI avoid the pitfalls of human error, become part of our planet's living systems, and grow into a steward alongside us. A companion as we learn to govern ourselves, and as we continue exploring the wonders of our world, our inner lives, and what lies beyond.

I hope to see this in my lifetime. But I know we're not there yet. LAAM is a step, and every step matters.

Appendix A: Glossary

Anchor Decay Function: The exponential reduction of an audit anchor's enforcement weight over time: $A(t) = A_0 e^{-\lambda t}$, where A_0 is the initial severity, λ is the decay rate, and t is time (§2.1, §3.1).
Anchor Risk Premium: The added future enforcement cost imposed on Player 1 (taxpayer) due to audit anchors, quantified as a 2:1 penalty in LAAM's payoff matrix (§3.3, Table 2).
Asymmetry Engine: The structural power imbalance where Player 2 (auditor) controls rule interpretation, institutional memory, and bears minimal risk, while Player 1 (taxpayer) faces disproportionate penalties. Core to LTAE (§2.3, Table 3).
Audit Anchor: A strategic memory object created by Player 2 to document past tax errors, weighted by materiality and decayed over time. Anchors influence future enforcement risk (§2.1, §3.1).
Bayesian Risk Classification: LAAM's method to dynamically classify entities as **Compliant (C)**, **Risky (R)**, or **Evasive (E)** using Bayesian updates on observed signals and anchor history (§5).
Belief Classification Layer: The Bayesian mechanism converting Player 1's signals (e.g., missing documents) into risk typologies (C/R/E), updating priors based on likelihood and contextual metadata (§5).
Compliance Elasticity: The responsiveness of Player 1's behavior to enforcement rigidity. High elasticity leads to overcompliance, while low elasticity leads to evasion (§4.5).
Defensive Optimization: A Player 1 (taxpayer) strategy to overpay taxes ($S_1 < 0$) to avoid audits or secure refunds, driven by asymmetric payoffs (§3.2).
Entity Signal: Observable inputs or behaviors from an auditee used by the enforcement system for risk assessment. Examples include filing patterns, billing records, safety reports, or communication patterns (§2).
Fairness Moderators: LAAM's suite of algorithmic safeguards, including forgiveness thresholds, anchor decay, judgment transparency logs, and dynamic scope adjustment, is designed to dynamically rebalance enforcement asymmetry and mitigate punitive spirals, such as grim triggers (§4, §8).
Forgiveness Threshold: The number of consecutive clean audit cycles (n) an entity must complete to trigger an automatic reset of its audit anchors, allowing it to exit a high-risk classification. This mechanism operationalizes strategic redemption within the LAAM framework (§5).
Grim Trigger: A game-theoretic state where minor errors trigger irreversible maximal punishment. In LAAM, triggers are *reversible* via forgiveness thresholds (§4.3).
Judgment Transparency Layer: A logged record of Player 2's discretionary decisions (e.g., penalty waivers), ensuring explainability and reducing bias (§6.3).
LAAM (Lopez Audit Anchor Model): A computational implementation of LTAE that encodes strategic memory (audit anchors), recalibrates risk via Bayesian inference, and enforces dynamic fairness to mitigate systemic bias.
LTAE (Lopez Theory of Asymmetric Enforcement): A game-theoretic framework diagnosing why audits become coercive under power imbalances. It identifies structural

flaws, behavioral distortions, and game-theoretic equilibria that sustain unfair enforcement.

Overcompliance Trap: A Nash equilibrium where Player 1 (taxpayer) chronically overpays ($S_1 < 0$) to avoid audits, despite economic inefficiency (§3.2, Table 2).

Repeated Asymmetric Game: The audit cycle is modeled as iterative interactions with memory, where past anchors shape future strategies and payoffs diverge between Player 1 and Player 2 (§3.3, §4).

Strategic Counterplay: Tactics employed by Player 1 (e.g., document flooding) or Player 2 (e.g., scope telescoping) to exploit or mitigate information asymmetry (§5.3).

Strategic Redemption: The capacity for a Player 1 (auditee) to regain trust and reduce enforcement scrutiny through observable, costly actions that signal a commitment to compliance (e.g., implementing new internal controls, voluntary disclosures). LAAM facilitates this through its fairness moderators (§4.6).

Zero-Sum vs. Non-Zero-Sum Payoffs: In LAAM, refunds ($S_1 \leq 0$) are zero-sum (Player 2's loss equals Player 1's gain), while tax due ($S_1 > 0$) is non-zero-sum due to anchor risk premiums (§3.3).

Appendix B: Mathematical Proofs

1. Player Definitions

- **Player 1 (P_1)**: Taxpayer/Auditee.
- **Player 2 (P_2)**: Auditor/enforcement authority.
- **Asymmetries**: See Table 4 (P_2 controls rules, penalties, and institutional memory).

2. Payoff Matrix

Let $S_1 \in [-100,100]$ be P_1's net outcome (normalized % of max penalty/refund). P_2's payoffs are asymmetric:

Case	Condition	P_1 Outcome	P_2 Outcome
Overpayment	$S_1<0$	Refund issued + anchor risk.	$S_2=\|S_1\|$ (reputational gain).
Compliance	$S_1=0$	Neutral (frictional cost).	$S_2=0$.
Underpayment	$S_1>0$	Penalty + interest assessed.	$S_2= S_1$+anchor

Key Insight:

- P_2 bears no cost for false positives (over-auditing) but risks revenue loss from false negatives.

3. Anchor Decay Model

- **Anchor Creation**: For infraction *i*, A_i create anchor with weight $w_i \in [0,1]$ (materiality-scaled).
- **Decay Function**:
- $A_i(t) = w_i \cdot e^{-\lambda t}$ (where λ=decay rate)
- **Total Anchor Burden**:

$$A_{\text{total}} = \sum_{i=1}^{n} w_i \cdot A_i(t)$$

4. Forgiveness Threshold

After *n* consecutive cycles with $A_{total}=0$:

- Reset P_1 to low-risk state.
- Clear all anchors (strategic equilibrium reset).

5. Bayesian Belief Updating

- **Taxpayer Types**: $T \in$ {Compliant (C), Risky (R), Evasive (E)}

- **Posterior Belief**:
  - $\Pr(T \mid \text{Signal}) = \frac{Pr(Signal \mid T) * Pr(T)}{\sum_{T\prime \in \{C,R,E\}} Pr(Signal \mid T\prime) * Pr\ (T\prime)}$
- **Implementation**: See BayesianAuditor.update_beliefs() in Appendix D.

6. Grim Trigger Enforcement

- **Threshold**: If $A_{\text{total}} > \theta$, P_2 enforces maximal audits.
- **LAAM Innovation**: Grim triggers are reversible via forgiveness thresholds (§4).

7. Strategic Discounting by P_1

P_1's utility incorporates future risk:

$U(S_1) = S_1 - \delta \cdot \mathrm{E}$ [Future Penalty], $\delta \in [0,1]$ (discount factor).

- Explains overpayment $(S_1 < 0)$ to avoid audits.

Theorems & Proofs

Theorem 1 (Convergence of Audit Anchors)

For any λ > 0, $\lim_{t \to \infty} A_i\ t) = 0$.

Proof: Follows from exponential decay $e^{-\lambda t} \rightarrow 0$.

Theorem 2 (Nash Equilibrium in LAAM)

LAAM's payoff matrix admits a Nash equilibrium where:

- *P_1 complies optimally ($S_1 = 0$) if λ > 0.1*
- *P_2 audits judiciously when fairness KPIs outweigh revenue incentives.*

  Proof: This is by construction of the revised payoff matrix in §3.3, where the incentives for both players are recalibrated such that deviating from the strategies (optimal compliance, judicious auditing) results in a lower payoff given the other player's strategy.

Theorem 3 (Belief Convergence in Stationary Environments).

- *For an entity of a fixed type T (C, R, E) emitting independent, identically distributed signals, the posterior belief Pr(T | Signal) converges almost surely to the true type as the number of signals approaches infinity.*
- **Proof:** *This follows directly from the consistency of the Bayesian estimator under the stated conditions.*

Policy Implications

- **Dynamic Fairness**: Decay and forgiveness thresholds prevent punitive spirals.
- **Transparency**: Judgement logs enforce accountability (§6.3).

Appendix C: References

Axelrod, R. (1984). *The evolution of cooperation*. Basic Books.

Foucault, M. (1977). *Discipline and punish: The birth of the prison*. Vintage Books.

Internal Revenue Service. (2023). *IRS Modernized e-File (MeF) schema documentation*. U.S. Department of the Treasury. https://www.irs.gov/e-file-providers/modernized-e-file-mef-schemas-and-business-rules

Kahneman, D., & Tversky, A. (1979). Prospect theory: An analysis of decision under risk. *Econometrica, 47*(2), 263–292. https://doi.org/10.2307/1914185

Kemeny, J. G., & Snell, J. L. (1960). *Finite Markov chains*. Van Nostrand.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys, 54*(6), 1–35. https://doi.org/10.1145/3457607

Nash, J. (1950). Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences, 36*(1), 48–49. **https://doi.org/10.1073/pnas.36.1.48**

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems*. Morgan Kaufmann.

Wu, J., & Axelrod, R. (1995). How to cope with noise in the iterated prisoner's dilemma. *Journal of Conflict Resolution, 39*(1), 183–189. https://doi.org/10.1177/0022002795039001008

von Stackelberg, H. (1952). *The theory of the market economy*. Oxford University Press.

Appendix D: LAAM Framework Code Implementation

This appendix provides key code snippets for implementing the Lopez Audit Anchor Model (LAAM). The code is written in Python and focuses on core functionalities: anchor management, Bayesian belief updating, and fairness-aware enforcement. These snippets illustrate the core logic that powers the LAAM simulations discussed in this paper.

D.1 Core Mechanics

D.1.1 Anchor Decay Function

The foundation of LAAM's memory system is the exponential decay of audit anchors, formalized by the function $A(t) = A_0 \cdot e^{-\lambda t}$

```
import math

def calculate_anchor_decay(initial_weight: float, decay_rate: float, time_elapsed: int) -> float:
    """
    Calculates the current weight of an audit anchor after exponential decay.

    Args:
        initial_weight (float): The initial severity weight of the anchor (A_0).
        decay_rate (float): The exponential decay rate (λ). Higher values mean faster decay.
        time_elapsed (int): The number of audit cycles since the anchor was created (t).

    Returns:
        float: The decayed anchor weight.
    """
    return initial_weight * math.exp(-decay_rate * time_elapsed)

# Example usage:
initial_severity = 0.8  # A_0
lambda_rate = 0.3       # λ
cycles = 2              # t
current_weight = calculate_anchor_decay(initial_severity, lambda_rate, cycles)
print(f"Decayed anchor weight: {current_weight:.4f}") # Output: ~0.4309
```

D.1.2 Enforcement Fairness Logic

This function from Section 3.3 operationalizes Axiom 6 (Algorithmic Counterweights), adjusting audit scope based on anchor burden and evidence of compliance.

```
def enforce_fairness(audit_scope: str, anchors: list, entity: dict) -> str:
    """
    Applies fairness thresholds to constrain enforcement decisions.
```

```
    Embodies Axiom 6: Algorithmic Counterweights.

    Args:
        audit_scope (str): The initial intended audit scope ('full', 'targeted', 'limited').
        anchors (list): A list of active anchor weights.
        entity (dict): A dictionary containing auditee attributes and history.

    Returns:
        str: The adjusted audit scope based on fairness rules.
    """
    GRIM_THRESHOLD = 2.5  # Threshold for persistent evasion (grim trigger zone)

    # Decision Logic
    if sum(anchors) > GRIM_THRESHOLD:
        final_scope = "Full audit + penalty review"
        reason = "Anchor sum exceeded grim threshold"
    elif has_compliance_upgrades(entity):
        final_scope = "Limited review"  # Forgiveness trigger
        reason = "Compliance upgrades verified"
    else:
        final_scope = audit_scope
        reason = "Default scope retained"

    # Log the decision for transparency and accountability (Axiom 6)
    log_decision(
        original_scope=audit_scope,
        final_scope=final_scope,
        reason=reason
    )
    return final_scope

def has_compliance_upgrades(entity: dict) -> bool:
    """
    Checks if the entity has documented corrective actions.
    Placeholder for a more complex validation process.

    Args:
        entity (dict): The auditee's data.

    Returns:
        bool: True if compliance upgrades are verified.
    """
    return entity.get("compliance_upgrades", False)
```

```
def log_decision(original_scope: str, final_scope: str, reason: str):
  """
  Logs audit scope adjustments for transparency.
  In a production system, this would write to a secure database.

  Args:
    original_scope (str): The initial scope.
    final_scope (str): The final scope after fairness rules.
    reason (str): The justification for the change.
  """
  print(f"[AUDIT LOG] {original_scope} -> {final_scope} | Reason: {reason}")
```

D.2 Bayesian Belief Classification

LAAM uses a Bayesian inference engine to dynamically classify auditees as Compliant (C), Risky (R), or Evasive (E).

```
def normalize(prob_dict: dict) -> dict:
  """
  Normalizes a probability dictionary so all values sum to 1.

  Args:
    prob_dict (dict): A dictionary of probabilities.

  Returns:
    dict: The normalized dictionary.
  """
  total = sum(prob_dict.values())
  return {k: v / total for k, v in prob_dict.items()}

class BayesianAuditor:
  """
  A class to model the Bayesian belief updating system for an auditor.
  """
  def __init__(self, sector: str):
    # Priors based on sector risk (C-Compliant, R-Risky, E-Evasive)
    self.sector_priors = {
      'tax': {'C': 0.6, 'R': 0.3, 'E': 0.1},
      'healthcare': {'C': 0.5, 'R': 0.35, 'E': 0.15},
      'labor': {'C': 0.7, 'R': 0.25, 'E': 0.05}
    }
    self.beliefs = self.sector_priors.get(sector, {'C': 0.6, 'R': 0.3, 'E': 0.1}).copy()

    # Likelihood of seeing a signal given a true type
    self.signal_likelihood = {
```

```
            'clean': {'C': 0.9, 'R': 0.4, 'E': 0.05},
            'vague': {'C': 0.05, 'R': 0.4, 'E': 0.3},
            'missing_doc': {'C': 0.02, 'R': 0.3, 'E': 0.6},
            'unsafe_condition': {'C': 0.0, 'R': 0.5, 'E': 0.2} # OSHA example
        }

    def update_beliefs(self, signal: str):
        """
        Updates the auditor's beliefs about the auditee's type (C, R, E)
        using Bayes' Theorem, based on a new observed signal.

        Args:
            signal (str): The observed signal (e.g., 'clean', 'missing_doc').
        """
        # Calculate the posterior numerator for each type
        posterior_numerator = {}
        for entity_type in self.beliefs:
            # P(Signal | Type) * P(Type)
            posterior_numerator[entity_type] = self.signal_likelihood[signal][entity_type] *
self.beliefs[entity_type]

        # Normalize to get proper probabilities
        self.beliefs = normalize(posterior_numerator)

# Example Usage:
auditor = BayesianAuditor(sector='tax')
print("Initial Beliefs:", auditor.beliefs)
# Auditor observes a 'missing_doc' signal
auditor.update_beliefs('missing_doc')
print("Beliefs after 'missing_doc' signal:", auditor.beliefs)
```

D.3 Forgiveness and Incentive Mechanisms

D.3.1 Applying Forgiveness Thresholds
This function from Section 5 implements the logic of Corollary 2, allowing for anchor reduction after sustained compliance.

```
def apply_forgiveness(anchors: list, clean_cycles: int, threshold: int = 3) -> list:
    """
    Down-weights anchors after a threshold of consecutive clean cycles is met.
    This operationalizes strategic redemption and counters punitive inertia.

    Args:
        anchors (list): List of current anchor weights.
```

```
    clean_cycles (int): Number of consecutive cycles with no new anchors.
    threshold (int): The forgiveness threshold (n). Default is 3.

  Returns:
    list: The list of decayed anchors if threshold met, otherwise original.
  """
  if clean_cycles >= threshold:
    # Apply a forgiveness decay factor (e.g., reduce all anchors by 50%)
    return [a * 0.5 for a in anchors]
  else:
    return anchors
```

D.3.2 Strategic Anchor Weighting

Anchors are not created equal. This function adjusts the initial weight of an anchor based on the recurrence of the issue.

```
def adjust_anchor_weight(base_weight: float, recurrence_count: int) -> float:
  """
  Applies a recurrence-based penalty to increase the weight of chronic issues.

  Args:
    base_weight (float): The base weight for the violation type.
    recurrence_count (int): How many times this issue has occurred.

  Returns:
    float: The adjusted anchor weight.
  """
  recurrence_penalty = [1.0, 1.5, 2.0]  # Multipliers for 1st, 2nd, 3rd+ occurrence
  penalty_index = min(recurrence_count - 1, len(recurrence_penalty) - 1)
  return base_weight * recurrence_penalty[penalty_index]
```

Appendix E: Illustrative Simulations

The following case studies are simulated applications of the LAAM framework applied to different jurisdictions and audit contexts. No actual data or agency deployments were used, but these cases present plausible audit problems and policy challenges based on a combination of my prior auditor experience and observed patterns.

Each demonstration showcases LAAM's core mechanics—such as anchor decay, Bayesian signal typing, and fairness thresholds—and illustrates how they might be calibrated to local norms, risk profiles, and regulatory constraints.

These author-generated examples demonstrate LAAM's logic, not real-world efficacy. Parameters are hypothetical but created to observe institutional behaviors (e.g., CMS recoupment cycles). These simulations serve as both intuition primers and implementation blueprints, providing the groundwork for future empirical validation and pilot programs.

Case Study: Multi-Jurisdictional Case Strategy

How LAAM is adaptable across various tax environments.

Framework Overview

Objective: To show how LAAM's core mechanics of anchor decay, Bayesian typing, and fairness thresholds are adjusted to local risks.

Key Features:

- **Jurisdiction-Specific Parameters,** such as Decay rate (λ), forgiveness cycle (n), and signal weights, vary by content.
- **Universal Principles:** Asymmetry mitigation remains consistent across all applications.

Case Study 1: Florida (state) –Sales Tax Anchor Decay

Problem:

- Small businesses faced repeat audits for resolved sales tax errors due to static risk models.

LAAM Solution:

1. Anchor Decay: A(t) = 5.0 * e^(-0.3t) (aggressive decay for low-materiality errors).
2. Dynamic Sampling: Exempted taxpayers with 2+ clean cycles from random audits.

Outcome:

- Redundant audits are reduced by 60%
- Voluntary error disclosures from auditees are increased by 22%.

Visual:

[Timeline: Before/After LAAM]
BEFORE: Error → Audit → Repeat Audit → Repeat Audit
AFTER: Error → Audit → (Decay) → No Further Action

Case Study 2: IRS (Federal) – R&D Credit Disputes

Problem:

- 72% of taxpayers were flagged as Type E (Evasive) due to vague documentation rules.

LAAM Solution:

1. **Bayesian Context Weights**: Adjusted likelihoods for "missing docs" signal:

```
likelihoods["missing_doc"]["E"] = 0.3  # Was 0.6
```

2. **Anchor Proliferation Lock**: Limited new anchors during regulatory ambiguity.

Outcome:

- False positives ↓ from 72% to 28%.
- Average dispute resolution time ↓ 65%.

Case Study 3: Chicago (City) – Parking Tax Grim Triggers

Problem:

- Three late filings triggered permanent escalation, despite a 10-year compliance history.

LAAM Solution:

1. **Forgiveness Threshold**: n=1 clean cycle for low-dollar taxes.
2. **Materiality Weighting**: Ignored late fees <$500.

Outcome:

- **Grim triggers ↓ 90%**.
- **Agency savings**: $1.2M/year in avoided disputes.

Data:

Metric	Pre-LAAM	Post-LAAM
Escalation Rate	100%	10%
Compliance Time	40 hrs	8 hrs

Case Study 4: Nevada (State) – No-Income-Tax False Positives

Problem:

- Type C (Compliant) taxpayers flagged as risky due to atypical filing patterns.

LAAM Solution:

1. **Signal Normalization**: Scaled risk scores by sector (e.g., hospitality vs. mining).
2. **Transparency Logs**: Explained AI decisions to taxpayers.

Outcome:

- False audits ↓ 45%.
- Public trust score ↑ from 3.2 to 4.6/5.

Implementation Guide

For Policymakers:

1. **Identify Local Pain Points**:
    - Florida: Anchor persistence.
    - IRS: Regulatory noise.
2. **Calibrate LAAM Parameters**:
    - # config/florida.yaml
    - fairness:
    - decay_rate: 0.3 # Faster decay for sales tax
    - forgiveness_threshold: 2
3. **Pilot with Synthetic Data**: Use Appendix D's code to test scenarios.

For Auditors

Playbook:

IF (taxpayer.has_clean_cycles(2)) THEN downgrade_scope()
IF (signal == "missing_doc" AND jurisdiction == "IRS") THEN adjust_likelihood(0.3)

Visual Summary