Technical Architecture & Governance of TML Smart Contracts: A Deterministic Enforcement Layer for Ternary Moral Logic

1. Introduction: The limit of Binary Governance

The prevailing computational paradigm of the twenty-first century has been built upon the rigid foundation of Boolean logic. Since the days of Shannon and Turing, the digital world has been bifurcated into zero and one, false and true, prohibit and permit. In the realm of financial transaction processing and basic data storage, this bivalent structure is sufficient; a ledger entry is either valid or it is not. However, as blockchain technology evolves from simple value transfer to the orchestration of autonomous Artificial Intelligence (AI) agents and complex social governance, the binary constraint has revealed itself to be a critical failure point. It lacks the nuance required to process the "grey areas" of human ethics, leading to brittle systems that either execute harmful actions because they technically adhered to code, or freeze entirely when encountering edge cases they were not programmed to recognize.

Ternary Moral Logic (TML), as conceptualized by Lev Goukassian and formalized in the accompanying research literature, proposes a radical architectural shift: the introduction of a third valid state, the "Sacred Zero". This state represents neither approval nor rejection, but rather *epistemic hold*—a mathematically enforced pause mandated by ethical ambiguity. This report provides an exhaustive technical and operational analysis of implementing TML as a deterministic enforcement layer on EVM-compatible platforms. It deconstructs the software architecture required to translate high-level philosophical mandates—such as "No Log = No Action" and the "Goukassian Promise"—into immutable Solidity bytecode.

The objective of this analysis is not merely to describe a theoretical framework but to engineer a "Constitution of Code." We explore the implementation of a strict Tri-State Logic state machine (+1, 0, -1) that prevents "God Mode" administrative overrides, ensures storage efficiency through Merkle-batched "Always Memory" logs, and secures the system via a "Hybrid Shield" defense strategy. By binding AI execution to these immutable cryptographic proofs, the TML framework aims to resolve the "black box" liability problem, ensuring that every autonomous action is preceded by a verifiable, court-admissible Moral Trace Log.

1.1 The Philosophical Imperative as Engineering Constraint

In traditional software engineering, "ethics" is often relegated to the layer of policy or user guidelines—soft constraints that sit outside the execution environment. TML inverts this relationship by embedding the ethical logic directly into the execution layer. The "Sacred Zero" is

not a suggestion; it is a blocking state in the smart contract that halts execution until specific resolution criteria are met.
The system operates on three core axioms, referred to in the documentation as "The Three Voices":

1. **+1 (Permit):** A clear ethical approval. The system proceeds with execution immediately.
2. **-1 (Prohibit):** A clear ethical rejection. The system reverts the transaction and logs the refusal.
3. **0 (Sacred Zero):** A state of recognized complexity. The system enters a holding pattern, requiring higher-order resolution (human stewardship or enhanced verification) before proceeding.

This triadic structure moves beyond the "move fast and break things" ethos, replacing it with a "move deliberately and document everything" architecture. The engineering challenge lies in implementing this "pause" mechanism on a synchronous blockchain like Ethereum, where transactions are atomic. The solution, as detailed in this report, involves a **Dual-Lane Latency Architecture** that separates clear-cut decisions (Fast Lane) from ambiguous ones (Slow Lane/Epistemic Hold).

2. Core State Machine Architecture: Implementing Tri-State Logic

The fundamental building block of the TML system is the departure from bool (true/false) to enum or int8 (tri-state) for all critical decision gates. This section details the strict state machine logic that governs TML-compliant entities.

2.1 The Tri-State Ontology and EVM Representation

In the Ethereum Virtual Machine (EVM), storage is expensive, and logic must be gas-efficient. While conceptually TML uses +1, 0, and -1, the Solidity implementation requires careful type selection to optimize for gas and safety.

TML State	Semantic Name	Internal Value	Operational Semantics	EVM Implication
+1	**PERMIT**	int8: 1	**Proceed:** The action is cleared.	Transaction executes via CALL or DELEGATECALL. State passes to EXECUTED.
0	**SACRED ZERO**	int8: 0	**Pause:** Ambiguity detected.	Transaction suspends. State transitions to EPISTEMIC_HOLD. Event LogSacredZero emitted.

-1	**PROHIBIT**	int8: -1	**Refuse:** Harm or violation detected.	Transaction reverts or records a REFUSAL event. Execution is blocked.

The choice of int8 allows for signed integers, aligning perfectly with the -1, 0, +1 nomenclature. However, a custom enum is often safer for internal solidity logic to prevent out-of-bounds errors, mapping enum State { PROHIBIT, PAUSE, PERMIT } to the integer values during input/output interfaces.

2.2 Strict State Machine Enforcement

To enforce the "No God Mode" constraint—where no administrator can force a transaction through a -1 state—the state machine's transition logic must be immutable and hardcoded. The contract logic does not check *who* is calling the function, but *what* the logic dictates.
The following Solidity interface illustrates the structural enforcement of the Three Voices. This interface acts as the gateway for all TML-compliant actions.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

/**
 * @title ITMLCore
 * @dev The constitutional interface for Ternary Moral Logic
enforcement.
 * Enforces the Tri-State logic (+1, 0, -1) and the Always Memory
mandate.
 */
interface ITMLCore {
    // The Three Voices of Ethical AI
    enum Voice {
        PROHIBIT, // -1: Clear ethical rejection
        PAUSE,    //  0: The Sacred Zero / Epistemic Hold
        PERMIT    // +1: Clear ethical approval
    }

    // Event emitted when the Lantern is lit (Sacred Zero triggered)
    event LanternSignal(
        bytes32 indexed traceId,
        address indexed agent,
        string reasonURI,
        uint256 timestamp
    );

    // Event emitted when a decision is anchored
    event MoralTraceAnchored(
        bytes32 indexed traceId,
        Voice decision,
        bytes32 contentHash
    );

    /**
     * @dev The Primary Enforcement Function.
     * Must be called before any executive action.
     * Implements "No Log = No Action".
     * @param _traceHash The Keccak256 hash of the off-chain moral
log.
     * @param _actionPayload The intended function call data.
     * @return status The operational voice (Permit/Pause/Prohibit).
     */
    function evaluateAction(bytes32 _traceHash, bytes calldata
_actionPayload)
        external
        returns (Voice status);
}
```

2.3 The "No God Mode" Logic

A critical requirement of TML is the elimination of "God Mode". In traditional governance contracts, a TimelockController or a Multi-Sig wallet often retains the power to upgrade the contract or bypass logic in emergencies. TML explicitly forbids this for the core decision logic. The "Rules" (+1, 0, -1 definitions) are locked in code.

To implement this, the TML Core contract must be deployed with its administrative privileges burned or transferred to a restricted SmartContractTreasury (SCT) that only accepts specific, pre-programmed inputs.

Mechanism for Immutable Governance:

1. **Deployment:** The TMLCore contract is deployed.
2. **Configuration:** Initial parameters (e.g., oracle addresses, bond curves) are set.
3. **Renunciation:** The owner role is revoked or transferred to the 0x0 address for logic upgrades.
4. **Parameter Tuning:** Only specific parameters (like gas price thresholds or oracle timeouts) can be adjusted by the Stewardship Council, but the *logic flow* of the Tri-State

machine is immutable. The Council cannot turn a -1 into a +1 without a complete system migration (fork), which would be visible and verifiable by the "Hybrid Shield."

3. "Always Memory": The Immutable Logging Architecture

The maxim "No Log = No Action" is the operational backbone of the TML framework. It transforms transparency from a passive virtue into an active engineering constraint. The system is architected such that the execution of any significant function is *cryptographically dependent* on the prior existence of a Moral Trace Log.

3.1 Pre-Action Commitment Scheme

The "Always Memory" architecture utilizes a Commit-Reveal scheme adapted for high-frequency AI decision making. The AI agent must "commit" to its reasoning before it is allowed to "reveal" (execute) the action.
Step-by-Step Execution Flow:

1. **Off-Chain Generation:** The AI agent analyzes a situation. It generates a JSON-structured "Moral Trace Log" containing:
    - **Context:** The inputs and environmental data.
    - **Reasoning:** The logic path taken (e.g., "Utilitarian calculus suggests X, but Deontological constraint Y forbids it").
    - **Decision:** The proposed Voice (+1, 0, -1).
    - **Identity:** The agent's licenseId and signatures.
2. **Hashing:** The agent computes LogHash = Keccak256(MoralTraceLog).
3. **Anchoring:** The agent submits LogHash to the TMLCore contract via the anchorLog() function.
4. **Verification:** The contract verifies the signature and stores the hash in the AlwaysMemory registry.
5. **Execution:** The agent calls the target function. The target function contains a modifier that queries AlwaysMemory. If the LogHash is not present, the transaction reverts with Error: No Log.

This sequence ensures that a permanent, immutable record of *intent* exists before any *effect* can be realized. In a legal context, this creates a "reverse burden of proof" : if an AI causes harm, the victim does not need to prove negligence; the absence of a log *is* the proof of negligence. If the log exists, it is fully discoverable evidence.

3.2 Storage Efficiency: Merkle Batching and Pointer Architecture

Storing full text logs on Ethereum Mainnet is economically impossible due to gas costs (approx. $50-$100 per kilobyte). Therefore, TML utilizes a **Pointer Architecture** combined with **Merkle**

Batching.
Pointer Architecture: The blockchain stores only the "Fingerprint" (Hash) and the "Pointer" (URI).

- **Hash:** bytes32 logHash (32 bytes). Guarantees integrity.
- **URI:** string contentURI (IPFS/Arweave link). Guarantees accessibility.

Merkle Batching for High Throughput: For high-frequency trading or rapid-response AI, submitting a transaction for every log is too slow. TML agents aggregate decisions into a local Merkle Tree.

- **Batch Size:** e.g., 1,000 decisions.
- **Submission:** The agent submits only the MerkleRoot to the smart contract.
- **Verification:** When executing a specific action from that batch, the agent provides the MerkleProof (sibling hashes) to prove that the specific decision was part of the anchored root.

Comparative Data: Gas Optimization Strategies

Strategy	Write Cost (Gas)	Verification Cost	Latency	Use Case
Direct Storage	~180,000	~2,100	Low	Low-volume, high-value decisions (e.g., firing a weapon).
Hash Anchoring	~45,000	~2,100	Low	Standard commercial AI operations.
Merkle Batching	~65,000 (per batch)	~30,000 (proof)	Medium	High-frequency trading, micro-transactions.

This tiered approach ensures that TML can scale from singular, high-stakes decisions to millions of micro-decisions without clogging the network, satisfying the "Storage Efficiency" constraint of the prompt.

3.3 The "Mandated Corpora" and Contextual Logging

TML requires not just a log of the decision, but a reference to the *ethical standard* used. This is referred to as the "Mandated Corpora". The framework integrates pointers to specific human rights treaties (e.g., UDHR, Geneva Conventions) and environmental standards (e.g., Planetary Boundaries).
Implementation: The TMLCore contract maintains a mapping(bytes32 => bool) public validCorpora registry.

- When an AI submits a log, it must reference the corpusId it is complying with.

- If the referenced corpus is not in the validCorpora registry (e.g., an outdated or rejected ethical standard), the system treats the input as invalid (-1).
- This allows the Stewardship Council to update the "Moral OS" by adding new treaties or standards to the registry, propagating these updates to all AI agents immediately.

4. The Goukassian Promise: Technical Artifacts

The "Goukassian Promise" is the ethical constitution of the framework, comprising three distinct artifacts: The Lantern, The Signature, and The License. While these sound symbolic, they are implemented as rigorous cryptographic and interface standards.

4.1 The Lantern (Proof of Hesitation)

The Lantern is the artifact that visualizes the "Sacred Zero." In the Solidity layer, this is not a UI element but a specific event emission pattern that signals *deliberation* to the network.
Technical Spec:

- **Trigger:** When evaluateAction returns Voice.PAUSE (0).
- **Mechanism:** The contract emits event LanternSignal.
- **Data:** Includes reasonURI (why the pause occurred) and releaseTime (minimum duration of the pause).
- **Frontend Integration:** DApps and wallets watch for this event. When detected, the UI locks the "Submit" button and displays the "Lantern" icon, informing the user that the AI is "thinking" or "consulting." This transforms the pause from a "system hang" into a "moral feature".

4.2 The Signature (Chain of Provenance)

The Signature ensures that every log is undeniably linked to a specific identity and version of the TML framework.
Technical Spec:

- **Structure:** A cryptographic signature (ECDSA) over the LogHash using the AI's private key.
- **Metadata:** Must include the ChainID, ContractAddress, and FrameworkVersion.
- **Non-Repudiation:** This prevents an AI operator from claiming "the algorithm did it." The operator's key signed the specific decision logic, binding them to the outcome.

4.3 The License (Covenant Against Misuse)

The License is implemented as a **Soulbound Token (SBT)**. It is a non-transferable NFT held by the AI agent's wallet.
The "No Spy / No Weapon" Enforcement:

- **Constraint:** The evaluateAction modifier checks IERC721(LicenseContract).balanceOf(msg.sender) > 0.
- **Revocation:** The License Smart Contract contains a revoke(address agent, string reason) function.
- **Authority:** Only the Stewardship Council (via multi-sig) can call revoke.
- **Logic:** If the License is revoked (burned), the TML Core automatically defaults all evaluateAction calls from that agent to -1 (PROHIBIT).
- **Effect:** This effectively "bricks" the AI's ability to interact with TML-compliant systems if it is found to violate the "No Spy" or "No Weapon" mandates , converting a legal breach into a functional lockout.

5. The Hybrid Shield: Dual-Layer Defense Architecture

The "Hybrid Shield" serves as the immune system of the TML framework. It is designed to protect the integrity of the moral history against corruption, fork attacks, or 51% attacks. It operates on two distinct layers: the Internal (Technical) Shield and the External (Anchoring) Shield.

5.1 Layer 1: The Technical Shield (Hash-Chain Integrity)

This layer exists within the smart contract state itself. It enforces a strict cryptographic dependency between sequential logs.
Mechanism: The Blockchain of Moral History Each AI agent has its own "Moral Chain" tracked by the contract.

- LastLogHash: The hash of the most recent decision.
- NextLogHash = Keccak256(NewDecision + LastLogHash).
- **Validation:** When submitting a new log, the contract verifies that the PreviousHash provided matches the stored LastLogHash.
- **Result:** This creates an unbreakable sequence. An attacker cannot insert a retroactive justification for a past action because it would invalidate the hash chain for all subsequent actions. This strictly enforces the linearity of moral reasoning.

5.2 Layer 2: The Anchoring Shield (Multi-Chain Redundancy)

To protect against platform-specific risks (e.g., an Ethereum reorg or censorship), the Hybrid Shield mandates **Multi-Chain Anchoring**.
The Anchoring Protocol: The system automatically broadcasts the root hashes of decision batches to multiple decentralized ledgers, leveraging the unique security properties of each.

Ledger	Role	Property Leveraged

Ethereum	**Execution Layer**	Turing-complete logic processing; state management.
Bitcoin	**Permanence Anchor**	Proof-of-Work immutability. Hashes written via OP_RETURN. High cost, extreme security.
Polygon/L2	**Speed Anchor**	High throughput, low cost. Captures high-frequency decision roots.
OpenTimestamps	**Time Anchor**	RFC 3161 compliance. Proves *when* a decision was made independent of block times.

Operational Flow:

1. The TMLCore contract accumulates a batch of Log Hashes.
2. Every N blocks (e.g., 6 hours), a "Keeper" bot triggers the anchorToExternalChains() function.
3. The contract emits an event with the MerkleRoot.
4. External "Bridge Oracles" (e.g., Chainlink or specialized TML nodes) pick up this root and write it to Bitcoin and Polygon.
5. The Oracle returns the TransactionID from the external chain to the TML Core.
6. The TML Core stores these ExternalProof IDs, creating a cross-referenced "Shield" where the truth is replicated across independent consensus mechanisms.

This structure makes it effectively impossible to erase a Moral Trace Log without simultaneously compromising Bitcoin, Ethereum, and Polygon—a scenario with a cost prohibitive to any actor, including nation-states.

6. The Sacred Zero: Operationalizing the Epistemic Hold

The "Sacred Zero" (State 0) is the most complex component of the TML architecture. It represents the "Epistemic Hold" —a state where the system recognizes it *does not know* the correct answer and therefore must pause.

6.1 The Dual-Lane Latency Architecture

To handle the Sacred Zero without freezing the entire blockchain, TML employs a Dual-Lane system.

1. **Fast Lane (+1 / -1):**
   - **Input:** Clear ethical parameters (e.g., "Donation to verified charity").
   - **Logic:** evaluateAction returns PERMIT.
   - **Outcome:** Atomic execution in the same block.
2. **Slow Lane (0):**

    - **Input:** Ambiguous parameters (e.g., "Purchase of dual-use technology").
    - **Logic:** evaluateAction returns PAUSE.
    - **Outcome:**
        - The transaction to *execute* the action is reverted/cancelled.
        - A *new* transaction is generated to create a "Pause Record" in the contract.
        - The system enters EPISTEMIC_HOLD for that specific ContextID.

6.2 The Epistemic Hold Protocol

Once a Pause is triggered, the system enforces a mandatory "Cool-down" and review period.
State Variables:

- mapping(bytes32 => PauseRecord) public pauses;
- struct PauseRecord { uint256 unlockTime; bool resolved; Voice resolution; address resolver; }

Sequence Diagram (Textual Representation):

1. **Trigger:** Logic detects conflict (e.g., Utilitarian Score > 0.8 but Deontological Flag = TRUE).
2. **Lock:** Contract creates PauseRecord. unlockTime = block.timestamp + MIN_PAUSE_DURATION (e.g., 24 hours).
3. **Signal:** LanternSignal emitted.
4. **Review:**
    - *Automated:* Oracles query additional data sources (e.g., "Check conflict zone map").
    - *Human:* Stewardship Council members receive a notification to review the case.
5. **Resolution:**
    - A Steward calls resolvePause(traceId, Voice.PERMIT).
    - The contract verifies the Steward's authority.
    - The PauseRecord is updated to resolved = true.
6. **Re-Execution:** The AI agent can now resubmit the transaction. The evaluateAction function sees the valid PauseRecord resolution and allows the action to proceed (+1).

This mechanism technically enforces the "Wait and See" approach, preventing high-speed algorithmic trading or autonomous weapons from acting on uncertain data.

7. Governance Structure: The Triangle of Trust

The governance of the TML ecosystem is designed to be "sovereign-grade," modeling a separation of powers similar to democratic governments but enforced by code.

7.1 The Technical Council (The Legislative)

- **Role:** Defines the cryptographic standards and maintains the codebase.
- **Composition:** Cryptographers, Solidity developers, System Architects.
- **Power:** Can propose upgrades to the *optimization* layer (e.g., making storage cheaper) but *cannot* alter the Tri-State logic definitions.
- **Check:** Any code upgrade requires a time-lock and a veto opportunity from the Stewardship Council.

7.2 The Stewardship Custodians (The Judicial)

- **Role:** The "Human in the Loop" for Sacred Zero resolutions. They act as the "Supreme Court" of the system.
- **Composition:** Representatives from Human Rights organizations (e.g., Amnesty), Environmental groups (e.g., Indigenous Environmental Network), and Legal Ethics bodies.
- **Power:** Can issue resolvePause transactions and revokeLicense transactions.
- **Check:** They cannot *initiate* code changes or access the Treasury funds directly.

7.3 The Smart Contract Treasury (The Executive)

- **Role:** The automated financial backbone.
- **Nature:** A decentralized autonomous vault. "A vault with no human door."
- **Funding Sources:**
    - **License Fees:** Paid by corporations using TML.
    - **Pause Deposits:** A "spam prevention" fee for triggering the Sacred Zero (refunded if the pause is deemed valid).
- **Disbursement Rules (Hardcoded):**
    - **Operational Costs:** Automatically pays gas fees for Anchor Oracles.
    - **The Memorial Fund:** mandates that a fixed percentage (e.g., 30%) of surplus funds is automatically routed to cancer research institutes (honoring Lev Goukassian's diagnosis) and victim compensation funds.
    - **Ecosystem Grants:** Remaining funds are released to the Technical Council only if system health metrics (uptime, anchor reliability) are met.

8. Economic Parameters and the "Moral Tax"

Implementing TML imposes a cost—verification is not free. This "Moral Tax" acts as a friction against reckless scalability.

8.1 Gas Cost Analysis

Operation	Standard Cost (Gas)	TML Cost (Gas)	Overhead	Components

Simple Transfer	21,000	~65,000	+210%	SLOAD (License check), KECCAK256 (Log verification).
Complex Logic	150,000	~220,000	+46%	Marginal cost decreases as base complexity rises.
Sacred Zero	N/A	~120,000	High	Event emission, struct storage, oracle request.

8.2 The Bonding Curve of Ethics

To prevent economic attacks (e.g., draining the Treasury via bogus Pause resolutions), the system uses a **Reputation Bonding Curve**.

- Stewards must stake tokens to participate in resolutions.
- If a Steward consistently votes against the consensus of the broader Council or is found to be bribed (via post-hoc analysis), their stake is slashed.
- This aligns the economic incentives of the "Judges" with the long-term integrity of the system.

9. Security Analysis and "No God Mode"

The most significant security feature of TML is the explicit removal of administrative override capabilities, known as "No God Mode".

9.1 The Immutable Core Pattern

Unlike upgradeable proxies (OpenZeppelin UUPS/Transparent), the core TML logic contracts should be deployed as **Immutable**.

- **Updates:** Updates are handled via a "Migration" pattern rather than a "Proxy" pattern. If the logic must change, a *new* contract is deployed, and users/agents must voluntarily migrate.
- **Implication:** This prevents a captured Technical Council from silently changing the rules for existing agents. The "Old Constitution" remains valid until users choose to move to the "New Constitution."

9.2 Attack Vector: Sybil Pausing

- **Attack:** An adversary generates thousands of AI agents to flood the system with "0" states, clogging the Stewardship Council's review queue.
- **Defense:**
    - **License SBT:** Only licensed agents can trigger a pause. Licenses strictly vetted (KYC/KYB).

    - **Pause Deposit:** Triggering a pause requires locking funds. If the pause is frivolous, the funds are forfeited to the Treasury.

9.3 Attack Vector: The "Lying Log"

- **Attack:** The AI logs "I am buying medicine" (+1) but the transaction payload executes "Buy weapons."
- **Defense: Binding Action Hashes**.
    - The evaluateAction function takes bytes calldata _actionPayload.
    - It hashes this payload: h = keccak256(_actionPayload).
    - It checks if h matches the intentHash stored inside the Moral Trace Log.
    - If they mismatch, the transaction reverts immediately. This cryptographically binds the *explanation* to the *execution bytes*.

10. Conclusion: Towards Sovereign Moral Architectures

The architecture detailed in this report represents a necessary evolution in the governance of autonomous systems. By translating the philosophical pillars of Ternary Moral Logic into the rigid constraints of EVM bytecode, we create a system where ethical behavior is not a choice, but a compiled requirement.

The integration of the "Sacred Zero" state acknowledges the limitations of machine certainty, creating a computational space for humility. The "Hybrid Shield" and "Always Memory" infrastructures ensure that this system is resilient against both external attacks and internal corruption. Finally, the "Goukassian Promise" serves as the binding covenant, ensuring that as AI scales in power and autonomy, it remains tethered to a verifiable, immutable history of moral reasoning.

This is not merely software; it is a digital constitution. In a world increasingly run by black-box algorithms, TML offers a blueprint for a "Glass House"—a system where every action is visible, every hesitation is documented, and every harm is accountable.

Sources Cited

Works cited

1. FractonicMind/TernaryMoralLogic: Implementing Ethical Responsibility in AI Systems - GitHub, https://github.com/FractonicMind/TernaryMoralLogic
2. The Day We Accidentally Became Disciples of a Dead Man's Digital Testament | by Lev Goukassian | Dec, 2025 | Medium, https://medium.com/@leogouk/the-day-we-accidentally-became-disciples-of-a-dead-mans-digital-testament-b83d96d46671
3. The Goukassian Promise. A self-enforcing covenant between… - Medium, https://medium.com/@leogouk/the-goukassian-promise-7abde4bd81ec
4. The Unbreakable Vow: How Ternary Logic's "Hybrid Shield" Protects from Corruption | by Lev Goukassian | Nov, 2025 | Medium,

https://medium.com/@leogouk/the-unbreakable-vow-how-ternary-logics-hybrid-shield-protects-from-corruption-1e6338d4744c

5. The Great Ternary Logic Incident: A Tale of Five Greeks and One Very Confused Document. | by Lev Goukassian | Dec, 2025 | Medium, https://medium.com/@leogouk/the-great-ternary-logic-incident-a-tale-of-five-greeks-and-one-very-confused-document-e621b95f06dd
6. Auditable AI by Design: How TML Turns Governance into ... - Medium, https://medium.com/@leogouk/auditable-ai-by-design-how-tml-turns-governance-into-operational-fact-37fd73e7b77e
7. FractonicMind/TernaryLogic: Ternary Logic enforces evidence based economics. It stops risky actions during uncertainty, records every decision with immutable proof, exposes hidden manipulation, anchors economic history across public blockchains, protects stakeholders from opaque systems, and ensures capital flows remain accountable to society and the planet. - GitHub, https://github.com/FractonicMind/TernaryLogic
8. TernaryMoralLogic/TML-VOLUNTARY-SUCCESSION.md at main - GitHub, https://github.com/FractonicMind/TernaryMoralLogic/blob/main/TML-VOLUNTARY-SUCCESSION.md
9. TernaryMoralLogic/TML_Pillars/Moral_Trace_Logs.md at main ..., https://github.com/FractonicMind/TernaryMoralLogic/blob/main/TML_Pillars/Moral_Trace_Logs.md
10. are the emerging corporate social disclosure laws capable of generating substantive compliance with human ri, https://www.publicacoes.uniceub.br/rdi/article/download/5355/3966
11. TernaryMoralLogic/Research_Reports/Perplexity Analysis of Lev Goukassian's Promise The Three Symbolic Cryptographic Safeguards.md at main - GitHub, https://github.com/FractonicMind/TernaryMoralLogic/blob/main/Research_Reports/Perplexity%20Analysis%20of%20Lev%20Goukassian's%20Promise%20The%20Three%20Symbolic%20Cryptographic%20Safeguards.md
12. So You Want to Build a Psychopath: A Sarcastic Guide to AI Liability | by Lev Goukassian, https://medium.com/@leogouk/so-you-want-to-build-a-psychopath-a-sarcastic-guide-to-ai-liability-bf62e943e99d
13. Applied Machine Learning for Analyzing and Defending against Side Channel Threats - UC Davis, https://escholarship.org/content/qt43v6p8c5/qt43v6p8c5.pdf
14. The Day the House Entered Epistemic Hold: A Story of Ternary Logic, Congress, and Credible Evidence | HackerNoon, https://hackernoon.com/the-day-the-house-entered-epistemic-hold-a-story-of-ternary-logic-congress-and-credible-evidence