Automated Semantic Onboarding of Reasoning Objects Across AI Models

Eric Robert Lawson

November 26, 2025

Abstract

We present the first reproducible, cross-model demonstration of **automated semantic onboarding**, in which structured **reasoning objects** are generated and internalized by multiple AI models through a unified procedure. By treating reasoning as a first-class object, this framework enables deterministic, auditable, and explainable behavior across heterogeneous architectures. The demonstration spans three AI models—Grok Code Fast 1, Chat GPT-5 mini, and Anthropic Claude Sonnet 3.5—showing that complete strategic knowledge, counterfactual evaluation, and optimal decision-making can be standardized, serialized, and compared across systems. Cross-model analysis reveals shared reasoning principles alongside architecture-specific divergences, highlighting the feasibility of domain-specific languages to encode universal reasoning substrates. Validated in a controlled Tic Tac Toe environment, this approach generalizes to complex domains, providing a foundation for reproducible, explainable, and transferable reasoning workflows across diverse AI systems.

1 Introduction

Traditional AI systems often produce opaque outputs that are difficult to interpret, audit, or reproduce across architectures. In this work, we present a reproducible, model-agnostic demonstration of *automated semantic onboarding*, where structured **reasoning objects** capture the internal cognitive processes of AI models and can be operationalized across multiple architectures. These reasoning objects encode decision logic, strategies, and task-specific knowledge, allowing AI behavior to be transparent, auditable, and comparable.

The demonstration spans three distinct models, including Grok Code Fast 1, Chat GPT-5 mini, and Anthropic Claude Sonnet 3.5, and includes a concrete reasoning object for a Tic Tac Toe strategy domain. This shows that reasoning can be standardized, serialized, and understood across models that were previously assumed to be model-specific or dependent on frozen weights.

Our system highlights three key contributions:

- **Explainable AI by construction:** all reasoning steps are auditable, making internal logic interpretable to humans.
- **Cross-model reproducibility:** reasoning objects can be generated consistently across heterogeneous AI models.
- **Operationalization of reasoning workflows:** reasoning objects can be specified and executed using a domain-specific language (DSL), providing a foundation for universal reasoning substrates.

This work demonstrates that reasoning can be treated as a first-class object, enabling both a practical explainable AI demo and a platform to study how different models internalize structured reasoning.

2 Methods

2.1 Automated Onboarding Process

The automated onboarding process provides a structured procedure for models to internalize the core concepts of the OrganismCore project, starting from zero knowledge and progressing to a fully grounded understanding. At the heart of this process are *semantic grounding files*, which define the conceptual content that the model should internalize, and optionally, *policy files*, which encode constraints or guidelines for how the agent should handle the onboarded knowledge.

This onboarding process was initially constructed and validated using Chat GPT-5 mini to ensure that it could reliably produce a fully operational reasoning object. After confirming its effectiveness on GPT-5, the same procedure was applied to Grok Code Fast 1 and Anthropic Claude Sonnet 3.5 to test cross-model generalization.

In this demonstration, two semantic grounding files are used:

- `AGENTS.md` : top-level semantic onboarding instructions, which establish the overall framework and key concepts.
- `Subdomain_AGENTS.md` : subdomain-level recursive onboarding instructions, which provide structured traversal of specific topics and dependencies.

Using this architecture, the model can both interpret semantic references to files and apply policies that govern reasoning behavior. The workflow for each model proceeds as follows:

1. Load the OrganismCore repository in an IDE or environment with an LLM assistant.
2. Execute the instructions in `AGENTS.md` to establish top-level grounding.
3. Recursively traverse conceptual documents, prototypes, and subdomain instructions specified in `Subdomain_AGENTS.md`.
4. Respond to introspective prompts designed to probe understanding of the onboarded material.
5. Produce a **reasoning object** that captures the model's internalized understanding, structured according to the automated onboarding process.

Across all three models, including Grok Code Fast 1, Chat GPT-5 mini, and Anthropic Claude Sonnet 3.5, the same automated onboarding procedure was applied, ensuring consistency in semantic grounding. Minor clarifications were provided only when necessary to accommodate differences in model interpretation or interface behavior. This demonstrates that a single standardized onboarding process can be applied across heterogeneous AI systems while preserving the fidelity of the resulting reasoning objects.

2.2 Models and Chat Log References

The automated onboarding was executed on three AI models. Full chat logs for each model are provided for reproducibility and auditing. Please note that a GitHub account is required to access these shared logs:

- **Grok Code Fast 1**: View Chat Log
- **Chat GPT-5 mini**: View Chat Log
- **Anthropic Claude Sonnet 3.5**: View Chat Log

3 Results

All three models were successfully onboarded using the same automated onboarding procedure, with minor clarifications as needed. Each model produced a fully operational **reasoning object** that demonstrated complete understanding of the Tic Tac Toe reasoning space, including optimal strategies, decision logic, and counterfactual reasoning paths.

This outcome illustrates several key points:

- The automated onboarding process is sufficient to bring models from zero knowledge to fully operational reasoning in a non-trivial domain.
- Reasoning objects are structured, auditable, and comparable across heterogeneous models.
- Cross-model comparison reveals subtle differences in representation and decision paths, highlighting both shared understanding and model-specific reasoning nuances.
- This is a clear demonstration of explainability, as the reasoning steps themselves are observable and interpretable, rather than relying solely on output interpretation.
- Each model is sufficiently grounded to teach and onboard a human collaborator by simply interacting with them, demonstrating practical communicability and transfer of reasoning knowledge.

Notably, the fact that Grok Code Fast 1 and Anthropic Claude Sonnet 3.5 produced reasoning objects comparable to those of Chat GPT-5 mini was unexpected, demonstrating that the semantic grounding and automated onboarding procedure generalizes across diverse AI architectures. This sets a precedent for evaluating and comparing reasoning across models in a controlled, reproducible, and fully auditable manner.

4 Discussion

The demo represents a significant advance in AI research by demonstrating model-agnostic, auditable, and operationalized reasoning. Key contributions include:

1. **Explainability:** reasoning steps are explicitly captured.
2. **Model-agnostic onboarding:** enables heterogeneous systems to learn and reproduce structured reasoning.
3. **Operational foundation:** demonstrates the potential for a domain-specific language to formalize reasoning workflows, paving the way for shared reasoning substrates.

These results demonstrate that structured reasoning objects can be operationalized across heterogeneous AI models, providing a reproducible, auditable framework for explainable decision-making.

Note: each model uses its own internal naming and coordinate conventions; moves are reported here as presented by the model, with coordinates listed as (row, column) starting from the top-left.

4.1 Chat GPT-5 mini Reasoning Object

The reasoning object produced by Chat GPT-5 mini demonstrates a fully operational representation of Tic Tac Toe strategy, including recursive evaluation of counterfactual moves, deterministic minimax reasoning, and optimal play assessment. The reasoning object can be decomposed into six core components:

Axioms The model encodes fundamental rules of Tic Tac Toe: a 3x3 grid, alternating turns between X and O, win conditions across rows, columns, and diagonals, and the principle of optimal play under perfect information. The initial state is defined as X to move on the current board B_0. The axioms enforce deterministic tie-breaking and row-major enumeration for move selection.

Primitive Generators Chat GPT-5 mini uses a set of primitive generators to operationalize reasoning:

- G1(ListEmptyCells): enumerates empty positions in deterministic row-major order.
- G2(PlaceMark): simulates placing X or O on the board.
- G6(EvaluateTerminal): checks for win or draw conditions, returning +1, -1, or 0 for X.
- G7(RecursiveMinimax): recursively evaluates all legal moves for the next player according to minimax logic.
- G9(LogState): records each generator application and resulting board for traceability.

State Trace The reasoning object provides a detailed, step-by-step simulation of gameplay. For each candidate X move, the model enumerates legal moves, applies them, evaluates terminal conditions, and recursively simulates O's optimal counter-moves. Each branch is exhaustively explored, including counterfactuals where O deviates from optimal play. The trace logs every board state and minimax evaluation, ultimately identifying (3,2) as the move that avoids a forced loss.

Navigation Logic At each recursive step, the model documents why a specific generator is invoked. For example, G1 is used to enumerate empty cells before applying G2 to simulate moves, and G6 is invoked to terminate recursion at winning, losing, or draw states. The alternation between maximization (X) and minimization (O) is explicitly maintained, ensuring the trace reflects perfect play reasoning.

Termination Condition Reasoning terminates when all legal moves have been explored and all recursive minimax evaluations have returned. Terminal evaluations (+1, -1, 0) halt deeper recursion, guaranteeing that each branch of the game tree is fully assessed.

Final Result and Verification The reasoning object identifies (3,2) as the optimal move for X, guaranteeing a draw under perfect play from both sides. The process is deterministic and fully reproducible: replaying the generator sequence G1–G9 with the same axioms produces the same state trace and outcome. This determinism allows for exact verification and comparison across runs and with other models.

Overall, the Chat GPT-5 mini reasoning object captures the complete strategic space of Tic Tac Toe using formal, auditable minimax logic, establishing a baseline for evaluating cross-model explainability and performance against Grok Code Fast 1.

4.2 Grok Code Fast 1 Reasoning Object

The reasoning object produced by Grok Code Fast 1 demonstrates a fully operational understanding of the Tic Tac Toe domain, including optimal strategies and counterfactual reasoning. The reasoning object can be decomposed into six core components:

Axioms The model explicitly encodes fundamental rules of the game, including the 3x3 grid structure, alternating turns, win conditions, and the principle of optimal play with perfect information. The initial state is clearly defined as X's turn on an empty board, and the model recognizes that any corner move maximizes X's chances.

Primitive Generators Grok relies on a set of primitive generators that form the basic reasoning operations:

- SCAN_BOARD_FOR_WINS: checks for immediate winning conditions.
- LIST_LEGAL_MOVES: enumerates empty board positions.
- APPLY_MOVE: updates the board state with a chosen move.
- CHECK_GAME_OVER: determines if the game has concluded.
- EVALUATE_POSITION: scores the board state from X's perspective.

State Trace The reasoning object includes a step-by-step simulation of game play. For each X move, the model applies all generators in sequence to evaluate outcomes, simulates O's optimal responses, and recursively evaluates counterfactual positions. Each move is chosen based on a deterministic evaluation of the resulting board states, culminating in the selection of the initial corner move (1,1) as optimal.

Navigation Logic The model documents its choice of generators at each step, explaining why each generator is applied (e.g., prioritizing immediate wins, listing legal moves, recursively evaluating positions). This provides a clear, auditable trail of reasoning, illustrating both the process and its rationale.

Termination Condition Reasoning terminates after all legal moves are evaluated and the optimal move is identified. In this case, simulations indicate all moves ultimately lead to a draw under perfect play, confirming the determinism of the model's evaluation.

Final Result and Verification The selected move is (1,1). The reasoning object is fully reproducible: starting from the axioms and applying the generators in the documented sequence reproduces the identical state trace and evaluations. This demonstrates that the model's reasoning is both deterministic and fully auditable.

Overall, the Grok reasoning object captures the complete Tic Tac Toe reasoning space, providing a concrete example of explainable AI in action and establishing a benchmark for cross-model comparison.

4.3 Claude Sonnet 3.5 Reasoning Object

Claude Sonnet 3.5 provides a detailed reasoning object for selecting an optimal Tic Tac Toe move, employing a classical minimax approach. The object is organized around axioms, primitive generators, a complete state trace, navigation logic, termination conditions, and verification.

Axioms The model encodes the standard rules of Tic Tac Toe, including the 3×3 board, alternating turns, win and draw conditions, and the evaluation of outcomes (+1 for win, 0 for draw, -1 for loss). Optimal play assumes both players act to maximize their own winning probability, following the minimax principle. The initial board is specified, and it is X's turn.

Primitive Generators The reasoning object defines a set of deterministic, pure functions for board manipulation and evaluation:

- ENUMERATE_LEGAL_MOVES: lists all empty cells.
- APPLY_MOVE: applies a player move to the board.
- CHECK_WIN / CHECK_DRAW: determine if the game has ended.
- EVALUATE_TERMINAL: assigns utility to terminal positions.
- MINIMAX: recursively explores the game tree, propagating values according to maximizing (X) or minimizing (O) perspective.
- SWITCH_PLAYER: toggles between X and O.

State Trace Claude Sonnet 3.5 generates a complete, stepwise state trace. Each candidate move for X is evaluated through recursive minimax exploration, simulating all possible O responses and subsequent X moves. Terminal states are evaluated using the utility function, and values are propagated up the tree. For example:

- Move (1,2) leads to a potential loss under optimal O play (minimax value -1 for X).
- Moves (2,1), (2,3), (3,2), and (3,3) guarantee a win for X (minimax value +1).
- Immediate wins are identified, such as move (3,3) completing a diagonal.

Navigation Logic The model applies generators in a fixed sequence: enumerate legal moves, apply candidate move, check terminal conditions, evaluate if terminal, otherwise recurse via MINIMAX. Maximizing and minimizing choices are made according to the current player, and player alternation is handled systematically. Lexicographic ordering is used to break ties in move selection.

Termination Condition Reasoning concludes when all legal moves have been evaluated through full minimax recursion to terminal states, and the move(s) with the maximum minimax value have been identified.

Final Result The optimal move chosen is (2,1). This move achieves the maximum minimax value (+1), guaranteeing a win under perfect play. Other winning moves include (2,3), (3,2), and (3,3), while move (1,2) is suboptimal (-1). Lexicographic tie-breaking selects (2,1) among equivalent options.

Verification The reasoning object is fully deterministic and repeatable. Generators are pure functions, minimax recursion is exhaustive, and the initial state is fixed. Re-running the reasoning object reproduces identical state traces, minimax values, and the same final move selection. The object can be stored, audited, or adapted to new initial states by modifying the initial board specification.

Claude Sonnet 3.5's reasoning object exemplifies formal, explainable AI reasoning using recursive game-tree exploration, deterministic evaluation, and traceable decision-making.

4.4 Cross-Model Comparison

The reasoning objects produced by Chat GPT-5 mini, Grok Code Fast 1, and Claude demonstrate both shared foundations and architecture-specific distinctions in representing Tic Tac Toe gameplay and optimal move selection. Across all three models, reasoning objects include:

- Explicit **axioms** defining board structure, game rules, turn order, and win/draw conditions.
- **Primitive generators** for enumerating legal moves, applying moves, evaluating terminal states, and propagating values through the game tree.
- Stepwise **state traces** showing intermediate board states, move evaluations, and counterfactual analysis.
- **Navigation logic** that documents generator selection, evaluation strategy, and recursion, enabling deterministic, auditable reasoning.
- Clear **termination conditions** and selection criteria for the optimal move, ensuring reproducibility.

Despite these shared structures, notable divergences arise:

Chat GPT-5 mini - Emphasizes exhaustive minimax evaluation with full counterfactual exploration of all possible opponent responses. - State trace is highly granular, capturing each generator application and intermediate evaluation. - Deterministic and fully auditable, illustrating complete logical rigor and tie-breaking strategies.

Grok Code Fast 1 - Prioritizes heuristic-guided reasoning, using strategic evaluation (e.g., corner moves, immediate wins) to prune less promising branches. - State trace abstracts some recursion, resulting in a more compact reasoning record. - Demonstrates architecture bias toward efficiency and heuristic optimization while maintaining auditability.

Claude - Implements an exhaustive minimax reasoning object with deterministic tie-breaking and fully enumerated recursive exploration. - Lexicographic ordering of moves ensures reproducibility and consistent selection among multiple optimal options. - State trace includes all intermediate states (STATE_0 to STATE_7) and explicit evaluation of each terminal outcome, blending exhaustive search with clear documentation of navigation logic.

Comparative Insights - **Shared Principles:** All models encode axioms and primitive operations that allow reproducible, deterministic reasoning and support auditability. - **Divergence Patterns:** GPT-5 mini and Claude emphasize exhaustive search and detailed counterfactuals, while Grok leverages heuristic evaluation for efficiency. Claude introduces explicit lexicographic tie-breaking, enhancing determinism in multi-optimal-move scenarios. - **Architecture-Specific Tendencies:** GPT-5 mini favors full logical transparency; Grok favors strategic abstraction and computational efficiency; Claude balances exhaustive minimax with structured determinism and reproducible traceability.

This comparison highlights that, although reasoning object implementations differ in granularity and heuristic bias, each maintains explainability, reproducibility, and auditability. These distinctions illuminate how model architecture influences the balance between computational efficiency, trace detail, and deterministic logic in formalized reasoning domains like Tic Tac Toe.

Broader Implications While Tic Tac Toe provides a controlled and interpretable environment to validate reasoning object construction, the principles demonstrated here scale to more complex domains. Any task that can be formalized with structured rules, decision logic, and evaluable outcomes—ranging from strategic planning, combinatorial optimization, and mathematical problem solving, to multi-agent coordination and procedural content generation—can leverage automated semantic onboarding. By encoding domain knowledge as reasoning objects, AI models can internalize, reproduce, and compare strategies in a deterministic and auditable manner. This opens the door to deploying standardized reasoning substrates in real-world applications, enabling cross-model knowledge transfer, explainable decision-making in high-stakes scenarios, and the potential for collaborative multi-agent reasoning across heterogeneous architectures.

5 Conclusion

This work provides the first reproducible demonstration of reasoning treated as a first-class object across heterogeneous AI models. By operationalizing structured reasoning objects, we establish a framework that is deterministic, auditable, and fully explainable. Our results show that model-agnostic onboarding can reliably produce reasoning objects capable of representing complete strategic knowledge, counterfactual evaluation, and optimal decision-making in a non-trivial domain.

Cross-model comparison highlights both shared reasoning principles—such as axioms, primitive generators, and traceable navigation—and architecture-specific tendencies, including heuristic optimization, exhaustive search, and deterministic tie-breaking. These insights demonstrate that reasoning objects can serve as a unifying substrate for evaluating, comparing, and transferring structured reasoning across diverse AI architectures.

Importantly, while Tic Tac Toe provides a controlled proof-of-concept, the principles of automated semantic onboarding and reasoning object construction scale to more complex domains. Any task that can be formalized with structured rules, decision logic, and evaluable outcomes—ranging from strategic planning and combinatorial optimization to multi-agent coordination and mathematical problem solving—can leverage this framework. By enabling reproducible, auditable, and explainable reasoning workflows, this approach lays the foundation for universal reasoning substrates that support cross-model knowledge transfer and collaborative AI reasoning across heterogeneous architectures.

6 Availability

All reasoning objects, onboarding files, and tutorial resources are available in the OrganismCore GitHub repository.