

From Gameplay Traces to Game Mechanics: Causal Induction with Large Language Models

Mohit Jiwatode[1(✉)], Alexander Dockhorn[2], and Bodo Rosenhahn[1]

[1] Institute for Information Processing, Leibniz University Hannover, Hannover, Germany
jiwatode@tnt.uni-hannover.de

[2] SDU Metaverse Lab, University of Southern Denmark, Odense, Denmark

Abstract. Deep learning agents can achieve high performance in complex game domains without often understanding the underlying causal game mechanics. To address this, we investigate Causal Induction: the ability to infer governing laws from observational data, by tasking Large Language Models (LLMs) with reverse-engineering Video Game Description Language (VGDL) rules from gameplay traces. To reduce redundancy, we select nine representative games from the General Video Game AI (GVGAI) framework using semantic embeddings and clustering. We compare two approaches to VGDL generation: direct code generation from observations, and a two-stage method that first infers a structural causal model (SCM) and then translates it into VGDL. Both approaches are evaluated across multiple prompting strategies and controlled context regimes, varying the amount and form of information provided to the model, from just raw gameplay observations to partial VGDL specifications. Results show that the SCM-based approach more often produces VGDL descriptions closer to the ground truth than direct generation, achieving preference win rates of up to 81% in blind evaluations and yielding fewer logically inconsistent rules. These learned SCMs can be used for downstream use cases such as causal reinforcement learning, interpretable agents, and procedurally generating novel but logically consistent games.

1 Introduction

Game AI has achieved remarkable milestones in the last decade [7,15,17,27]. Advances in Deep Reinforcement Learning (DRL) and other game AI frameworks have enabled superhuman performance in combinatorially complex domains, from traditional board games like Go and Chess [27] to high-dimensional video games like Atari and StarCraft II [15,17] and large-scale stochastic planning environments [26]. However, these systems largely operate as "black boxes", relying only on statistical pattern recognition and learning correlations to master a policy that maximizes a reward. They rarely possess an explicit, transferable representation of the game's causal mechanics. The fundamental question: "*Does the*

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
M. De Marsico et al. (Eds.): ICPR 2026, LNCS 16824, pp. 1–16, 2027.
https://doi.org/10.1007/978-3-032-31927-2_1

model understand ***why*** *an action leads to a specific outcome?*", remains largely unanswered. This lack of interpretability creates significant hurdles: agents struggle to generalize when rules shift slightly, and their decision-making processes are opaque to human users [5].

To address this, we argue for moving beyond correlation-based learning towards causal reasoning. Rooted in the framework of Structural Causal Models (SCMs), causal AI offers the tools to reason about interventions ("*What if I push this block?*") and counterfactuals ("*Would I have died if I hadn't jumped?*") [20]. In this work, we propose the General Video Game AI (GVGAI) framework [23] as a testbed for causal induction: the process of acquiring causal knowledge from experience [20]. GVGAI games are defined in the Video Game Description Language (VGDL) [25], a human-readable format that explicitly encodes game entities, interaction dynamics, initial state, and terminal conditions. We argue that VGDL is effectively a domain-specific Structural Causal Model; extracting it from raw observations is equivalent to discovering the causal laws of the game world.

While prior work has explored procedural content generation using Large Language Models (LLMs), we investigate a more fundamental capability: **Causal Induction**, i.e., "*Can an LLM observe a sequence of game states and reverse-engineer the underlying VGDL rule set?*" We introduce a novel evaluation framework that tests this capability through two lenses: a **Multi-Class Game Identification** task to assess semantic understanding, and a **VGDL Synthesis** task comparing direct code generation against a neuro-symbolic, SCM-mediated approach. The code and appendix can be found here: https://github.com/jiwatode-mohit/SCM_4_GVGAI

Our contributions are as follows:

- **Semantically Diverse Benchmark:** We construct a benchmark of 9 representative games from the GVGAI environment, identified via semantic clustering using Sentence-Transformers and K-Means to ensure rigorous testing across diverse game mechanics.
- **Self-Conditioned Evaluation:** We introduce a novel prompt-based evaluation for game identification that specifically tests an LLM's robustness against "hallucinated" or "memorized" mechanics versus actually understanding the game mechanics from observations.
- **SCM-Mediated Synthesis Framework:** We propose a dual-stream generation pipeline that forces LLMs to explicitly draft a causal graph (SCM) before generating code, evaluating whether intermediate causal reasoning improves the fidelity of synthesized game rules.

2 Background and Related Work

2.1 Causality and Structural Models

Causality extends beyond statistical association, enabling systems to predict the effects of actions (interventions) and reason about alternative scenarios (coun-

terfactuals). Judea Pearl [20] defines this hierarchy as the "Ladder of Causation," distinguishing between mere association, intervention (altering a variable's value), and counterfactuals (reasoning about hypothetical alternatives). At the core of this framework are Structural Causal Models (SCMs), defined as a triple (U, V, F), where endogenous variables V are determined by exogenous variables and noise U from structural functions F.

While Hammond et al. [11] have extended causal reasoning to game-theoretic models, empirical causal discovery (inferring F from raw data) remains a challenge. In our work, we treat the game engine as the ground-truth SCM and the VGDL script as its symbolic representation. Unlike Baier et al. [1], who use causal subgoals for score optimization, our objective is the reconstruction of the causal laws themselves.

2.2 VGDL as a Causal Proxy

The GVGAI framework is widely utilized as a testbed for evaluating agent generalizability, challenging algorithms to adapt to diverse, real-time arcade environments [23]. The framework employs the Video Game Description Language (VGDL) to define these environments through four modular sections: a `SpriteSet` for instantiating distinct game actor types and static objects, a `LevelMapping` that translates ASCII symbols into spatial sprite instances, an `InteractionSet` for governing rule-based event handling, and a `TerminationSet` for establishing win/loss criteria. We propose that the VGDL schema provides a natural mapping to the SCM formalism. Specifically, we posit that the `SpriteSet` corresponds to the set of endogenous variables V, the `LevelMapping` defines the exogenous initialization context U, and the `InteractionSet` explicitly encodes the structural equations F (e.g., `Avatar > Wall: stepBack`).

Prior attempts to generate VGDL have struggled with logical consistency. Nielsen et al. [18] used evolutionary algorithms but found the resulting games often "broken" or unplayable due to incoherent mechanics. More recently, Hu et al. [12] demonstrated that while LLMs like GPT can generate VGDL, they frequently "hallucinate" interactions that conflict with the game's physics. Our work addresses this by enforcing an intermediate SCM structure to ground the LLM's reasoning.

2.3 Neurosymbolic Reasoning with LLMs

While LLMs exhibit impressive fluency, they often lack robust logical reasoning and state tracking [4,10]. To mitigate this, recent "System 2" approaches integrate LLMs with symbolic solvers [8,19] or use Chain-of-Thought (CoT) prompting [31] to elicit intermediate reasoning steps.

In the domain of games, Todd et al. (GAVEL) [30] and Tanaka et al. [28] have explored using LLMs for Ludii game description language generation. However, Bateni et al. [2] highlight that models perform well on known games but fail on rule variations, reinforcing the need for a prompt setting to test whether a model

is reasoning causally or simply retrieving memorized rules. Furthermore, Li et al. [14] warn of "task contamination," where zero-shot capabilities are often the result of memorization, a critical consideration when using public benchmarks like GVGAI.

3 Methodology

To systematically evaluate LLM capabilities within the GVGAI domain, we structured our methodology into four distinct phases: (1) the construction of a semantically diverse benchmark, (2) the definition of the experimental setup, including model and data selection, (3) a classification task to assess game identification, and (4) a synthesis task to evaluate rule induction and code generation.

3.1 Benchmark Construction

The General Video Game AI (GVGAI) competition contains 80 distinct games, many of which are simple variants or share significant semantic overlap [23][1]. Rather than selecting games randomly or subjectively, we sought to create a benchmark that maximizes semantic diversity. To achieve this, we developed a three-stage pipeline: translating game logic to natural language, embedding it into a vector space, and clustering the vectors to identify a compact set of representative games.

Preliminary Model Selection. To select a language model capable of high-quality summarization and further tasks, a few state-of-the-art open-weight models were put through a classification test on a random subset of 10 games from a PhD thesis [6]. Qwen3-8B emerged as the best candidate LLM and was subsequently used for the VGDL translation task.

VGDL to Natural Language Translation. Raw Video Game Description Language (VGDL) files use symbolic code unsuitable for standard sentence embedding models, which are trained on natural language rather than structured logic [24]. Therefore, we translated VGDL files for the 80 selected games into concise natural language summaries (under 100 words) using **Qwen3-8B**. These descriptions provide coherent overviews of game mechanics and goals, making them suitable for embedding.

Vector Embedding. The resulting natural language descriptions require an embedding model optimized for sentence-level semantics. We employed Sentence-Transformers to convert each game description into a dense vector.

[1] A snapshot of the training set is archived at: https://web.archive.org/web/20190214020741/https://www.gvgai.net/training_set.php?rg=1.

Semantic Clustering. To identify distinct semantic groups within this space, we apply the K-Means clustering algorithm directly to the vector embeddings. The optimal number of clusters was determined by performing a hyperparameter sweep for the cluster count k ranging from 5 to 14. We conducted a comprehensive comparative analysis of various preprocessing strategies, including L2 normalization and Standard scaling, along with dimensionality reduction techniques like PCA [21] and UMAP [16], which is detailed in the appendix. However, for our final benchmarks, we utilized the raw embeddings. This approach preserves the original semantic distances and aligns with the standard implementation demonstrated in the official Sentence-Transformers documentation. The optimal value of k was found to be 9.

Consequently, we partitioned the 80 games into 9 distinct semantic clusters. The game closest to the centroid of each cluster was selected as the representative for that cluster, forming our 9-game benchmark. Figure 1 provides a visual projection of these clusters.

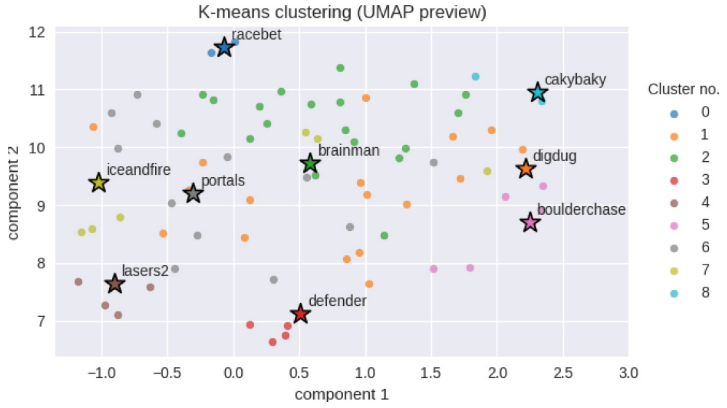


Fig. 1. 2D UMAP projection of the 80 games across 9 semantic clusters. The representative games are marked within their respective regions.

3.2 Experimental Setup

Following the identification of the 9 cluster representatives, we established a unified experimental environment for both classification and synthesis tasks.

Model Selection. While Qwen3-8B, a reasoning model, was selected for generation due to its preliminary success, we hypothesized that its performance might not be solely attributable to the architecture, but rather its underlying reasoning capabilities.

To investigate this, we expanded our model selection to include other models from the Qwen family to test architectural consistency. This diverse selection

allows us to decouple the effects of model architecture from general reasoning ability in the game identification task.

Observation Generation. Ten sets of observation sequences per game were generated in the GVGAI framework, with each sequence limited to ten frames to constrain the context window and ensure computational efficiency.

3.3 Task I: Multi-class Game Identification

This task evaluates whether Large Language Models (LLMs) can identify a game from short sequences of ASCII-grid gameplay traces by reasoning about underlying mechanics, rather than relying on memorized names or prompt artifacts.

Given one set of candidate **game descriptions** (expert-authored or generated/refined by the model from the prompts below) and **gameplay traces** with actions, the model must output exactly one game label.

To isolate how different sources of semantic information influence classification, we evaluate models under four prompt variants. We refer to this setup as self-conditioned evaluation, as models are tested using descriptions they generate or refine themselves.

$P_{Standard}$ **(Human Expert):** Fixed expert-written descriptions focusing on mechanics and objectives. This serves as a high-quality semantic baseline. Unlike the other variants, this is a non-generative control condition. While it follows the same pipeline structure for simplicity, the LLM acts effectively as a pass-through, preserving the exact expert wording.
P_{Cons} **(Constructive Refinement):** The model refines the expert descriptions before classification. This tests whether models can improve causal clarity.
P_{Dest} **(Destructive/Parametric):** The model generates descriptions from the game name alone. This removes external context and controls for memorization [2] and task contamination [14].
P_{VGDL} **(VGDL-Summarization):** The model summarizes the raw ground-truth VGDLs into natural language descriptions. This tests the model's code-to-text reasoning.

For each model, observation sequence, and prompt variant, we record the predicted label and classification accuracy. The complete workflow is illustrated in Fig. 2.

3.4 Task II: VGDL Synthesis Framework

The second analytical task utilizes a hierarchical context injection strategy to evaluate the capabilities of LLMs in synthesizing VGDL files directly from raw gameplay observations. The pipeline facilitates a comparative analysis between direct code synthesis and a causal-reasoning-mediated approach.

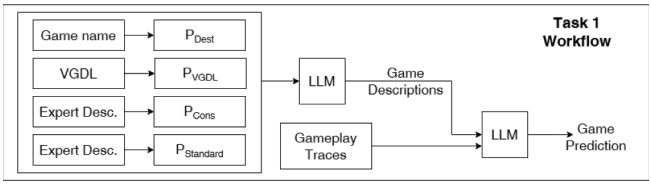


Fig. 2. Task 1 Workflow: The model predicts the game label based on a sequence of observations and a game description. The description is either fixed (Standard) or generated/refined by the LLM (Cons, Dest, VGDL) prior to classification.

Efficiency Trade-Off and Model Selection. To identify models suitable for this computationally intensive VGDL synthesis, we analyzed the trade-off between accuracy and logarithmic runtime (Fig. 3) for Task I. From the **Pareto Front** (red line), we selected two candidates that exceed 75% accuracy, balancing performance with feasibility: **Qwen3-8B** and **QwQ-32B (quantized)**.

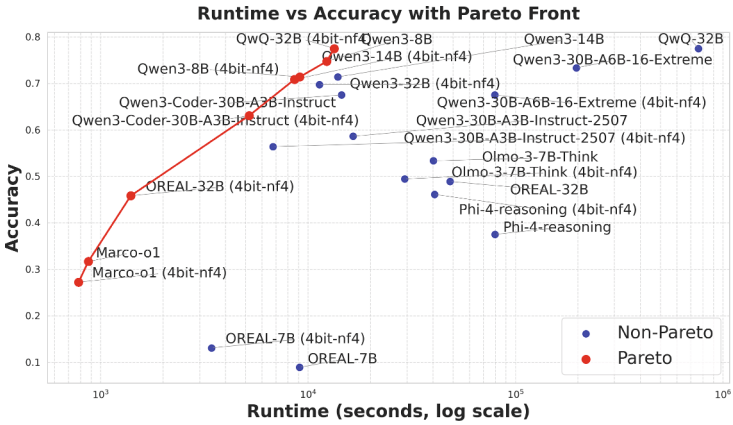


Fig. 3. Runtime vs. Accuracy with Pareto Front. The red line indicates the Pareto frontier. Qwen3-8B and QwQ-32B lie on the frontier, justifying their selection for the resource-intensive synthesis task. (Color figure online)

Hierarchical Context Engineering. To isolate the impact of information density on code generation, we define five cumulative levels of context (C_L). This extends the approach of Hu et al. [12] of using progressively informative prompt contexts to systematically isolate the effects of information. At each level, the model is tasked with generating a complete VGDL definition.

– **Level 0 (The Baseline):** The model is provided strictly with the raw grid observation string (ASCII matrices of the game state). No semantic descrip-

tions, grammar rules, or game names are included. This tests if the model can infer physics dynamics solely from visual state transitions.
- **Level 1 (The Syntax Check):** The prompt is augmented with the *Common VGDL Grammar* specification and a single example of a valid VGDL file from a game outside our benchmark testing set. This tests if providing valid vocabulary (e.g., `SpriteSet`, `InteractionSet`) allows the model to structure its physical inferences into valid code.
- **Level 2 (The Exact Game):** The context includes the syntax (Level 1) plus the target game's name and its specific high-level natural language description. This tests if the model can correlate natural language rules (e.g., "The avatar must avoid the goblin") with the visual grid entities.
- **Level 3 (The Distractor Test):** The input includes the observation and syntax, but instead of the single correct description, the model is provided with a dictionary of potential games (the target game plus distractors). This tests if the model can implicitly identify the correct game mechanics when presented with competing rule sets.
- **Level 4 (The Completionist):** The model is provided with a partial VGDL file where specific logic blocks have been removed. Specifically, the `SpriteSet` (entity definitions) and `LevelMapping` (layout) are preserved, while the `InteractionSet` (collision physics) and `TerminationSet` (win conditions) are completely removed. This tests if the model can reverse-engineer the missing interaction logic given the static entity definitions and visual evidence.

Dual-Stream Generation Architecture. For every context level L, the framework executes two parallel generation tasks to produce the final VGDL output. The inputs and constraints differ as follows:

Stream A: Direct Synthesis (Non-SCM):

- The input is the JSON-structured payload containing the accumulated context for Level L.
- The model acts as a "VGDL Architect." It is instructed to perform an analysis of the objects in the grid before directly outputting the raw, executable VGDL code block derived from pattern matching and context inference.

Stream B: Structural Causal Model (SCM) Guided:

- This stream utilizes the same context Level L but enforces a constrained Chain-of-Thought (CoT) approach defined by a strict **SCM Blueprint**. This approach is directly motivated by the findings of Hu et al. [12], who observed that LLMs frequently "hallucinate" incorrect game logic due to conflicts with natural language word order. Furthermore, Tanaka et al. [29] demonstrated that grammar constraints alone are insufficient for capturing correct mechanics without gameplay alignment. By enforcing an intermediate SCM representation, we provide the structured reasoning aid that Bateni et al. [3] identified as necessary for accurate rule simulation.

- **Step 1 (Causal Graph Generation):** The model first generates a structural causal model in JSON format. The graph nodes are strictly typed into layers: *Design* (e.g., ActionSpace), *Dynamics* (e.g., InteractionMechanics), and *Observation* (e.g., LevelEncoding).
- **Step 2 (Compiler Phase):** The generated SCM JSON is fed back into the model with instructions to "compile" the causal graph into executable VGDL code.

As illustrated in Fig. 4, the selected context payload is processed through two parallel streams: a direct synthesis approach (Stream A) and a causal-reasoning-mediated approach (Stream B).

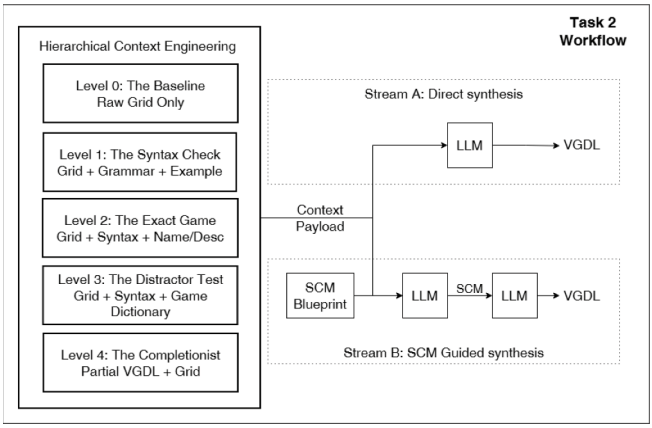


Fig. 4. Task II Workflow Diagram. This overview illustrates the hierarchical context engineering levels (left) feeding into the dual-stream generation architecture (right). Stream A utilizes direct synthesis, while Stream B enforces a Structural Causal Model (SCM) blueprint.

Evaluation Metrics. The fidelity of the generated VGDL is assessed in a post-processing phase using three distinct metrics:

- **VGDL Cosine Similarity:** The cosine similarity between the vector embeddings of the generated VGDL code and the ground truth VGDL file.
- **Semantic Text Similarity:** The cosine similarity between the natural language descriptions derived by translating both the generated and ground truth VGDL files using Qwen3-8B, assessing conceptual alignment.
- **LLM-based Preference Evaluation:** To capture qualitative alignment, three state-of-the-art closed-source models (GPT-5.2, Gemini 2.5 Pro, Claude 4.5 Opus), and the Qwen3-8B model are presented with the ground truth VGDL and the outputs from Stream A and Stream B. They are queried to perform a blind ranking of which stream's output better preserves the

semantic and logical integrity of the original game mechanics from the ground truth VGDL.

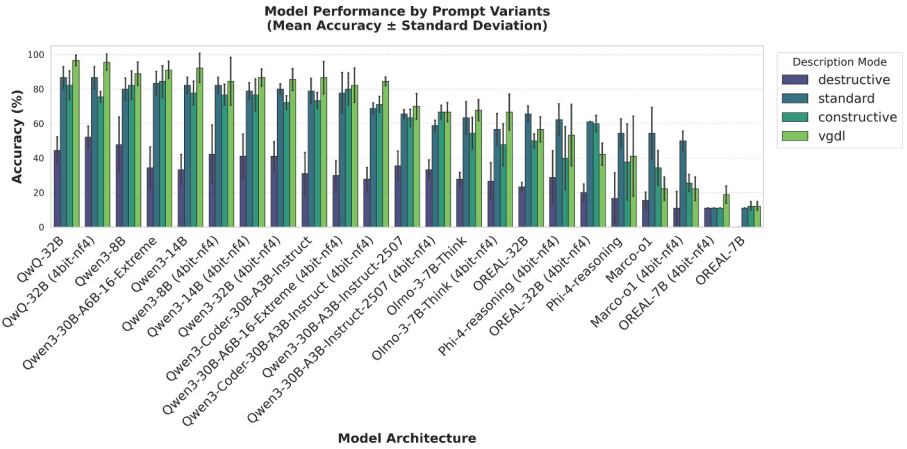


Fig. 5. Model Performance by Description Mode. The figure illustrates the mean accuracy across description types. The stark drop in the P_{Dest} trajectory compared to $P_{Standard}$ indicates that models lack complete internal knowledge of the games, relying heavily on provided context.

4 Results and Discussion

4.1 Task I: Multi-class Game Identification

We evaluated a suite of Large Language Models (LLMs) on their ability to identify games from ASCII grid observations across four distinct description prompt types ($P_{VGDL}, P_{Standard}, P_{Cons}, P_{Dest}$).

Overall Model Performance. The overall classification accuracy varied significantly across model architectures. As shown in Fig. 5, the **QwQ-32B (quantized)** model achieved the highest global average accuracy (≈77.5%) with a runtime of around 13300 s, followed closely by **Qwen3-8B** (≈74.7%) at a slightly lower runtime of around 12300 s. In contrast, other families of models, such as OREAL-7B and Marco-01, yielded near-random performance, highlighting the limitation of these models for understanding ASCII-based spatial reasoning.

Figure 5 further illustrates a critical finding regarding the models' internal knowledge. We observe a **stark drop in performance** when moving from the context-rich settings ($P_{Standard}$ or P_{VGDL}) to the destructive setting (P_{Dest}), where the model must rely on its training data to generate the game description.

Context Dependency: For top-performing models like Qwen3-8B, the accuracy drops from approximately 80% in the standard setting to roughly 45% in the destructive setting. This degradation supports the hypothesis presented in our methodology: despite their size, these models do not possess a robust understanding of the specific GVGAI game rules from their training data alone.

Validation of Methodology: The inability of models to maintain performance without external descriptions validates our use of the Destructive prompt as a control variable. It confirms that the models do not have a good understanding of the game mechanics.

Semantic Confusion Analysis. To validate the effectiveness of semantic clustering, we analyzed the global confusion matrix (Fig. 6). We observe strong diagonal performance (recall) for distinct representative games such as *racebet* (84%) and *portals* (66%).

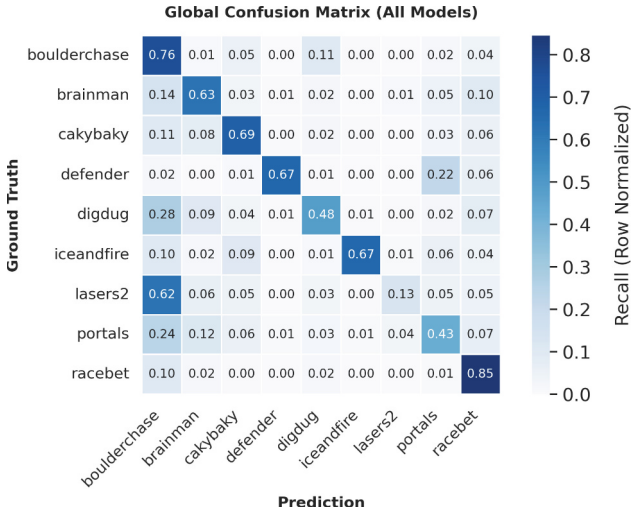


Fig. 6. Global Confusion Matrix aggregated across all models. High diagonal values indicate effective identification of archetypal games, though semantic bleed remains between mechanically similar titles like *digdug* and *boulderchase*.

However, consistent with the limitations in model understanding, confusions were largely contained within semantically similar clusters. For example, significant confusion exists between *digdug* and *boulderchase*. Both games involve grid-based digging and falling boulders, suggesting that while models can recognize general mechanical patterns from the ASCII grids, they struggle to distinguish the specific causal nuances and mechanics that define the exact game title. This may also be because the gameplay traces failed to capture the particular states and transitions that distinguish the different mechanisms.

4.2 Task II: VGDL Synthesis Results

We evaluated the fidelity of the generated Video Game Description Language (VGDL) mechanics using the dual-stream pipeline: Direct Synthesis vs. Structural Causal Model (SCM)-Guided, across 45 unique generation tasks. The evaluation compared **Qwen3-8B** against the quantized version of **QwQ-32B** model. Results were validated by a panel of four Large Language Model (LLM) judges (GPT-5.2, Gemini 2.5 Pro, Claude 4.5 Opus, and Qwen3-8B).

Overall Model Performance and Judge Agreement. As detailed in Table 1, the SCM-mediated approach (Stream B) was consistently preferred over Direct Synthesis (Stream A) across both models and all judges. For **Qwen3-8B**, the SCM stream achieved a dominant win rate of approximately **67%–81%** ($p < 0.01$, binomial test), with very few ties. In contrast, the larger **QwQ-32B** model showed a higher incidence of ties (up to 15.9% with Opus), suggesting that its "Direct" synthesis is more frequently comparable in quality to its SCM output than that of the smaller model. However, the preference for SCM remains statistically significant ($p < 0.01$, binomial test), peaking at **75%** with the Qwen-based judge.

Table 1. Overall Judge Win Rates comparing Direct Synthesis (NS) vs. SCM-Guided (SCM). The abbreviation **NS** denotes "Non-SCM" (Direct Synthesis). The reasoning model (QwQ-32B) exhibits higher tie rates, but SCM remains the preferred method across all judges.

Model	Judge	NS %	SCM %	Tie %
Qwen3-8B	GPT-5.2	31.0%	**66.7%**	2.4%
Qwen3-8B	Gemini 2.5 Pro	33.3%	**66.7%**	0.0%
Qwen3-8B	Claude 4.5 Opus	28.6%	**66.7%**	4.8%
Qwen3-8B	Qwen-3-8B	19.0%	**81.0%**	0.0%
QwQ-32B	GPT-5.2	22.2%	**64.4%**	13.3%
QwQ-32B	Gemini 2.5 Pro	40.0%	**60.0%**	0.0%
QwQ-32B	Claude 4.5 Opus	31.8%	**52.3%**	15.9%
QwQ-32B	Qwen-3-8B	17.5%	**75.0%**	7.5%

Context Level Sensitivity. Table 2 reveals a divergence in how the two models utilize context:

- **Induction (Levels 0-3):** Both models benefit significantly from the SCM pipeline in low-information settings. At Level 0 (Observation Only), Qwen3-8B's SCM output was preferred **91.7%** of the time. QwQ-32B also favored SCM (58.8%), though with a high tie rate of 17.6%, indicating that the larger model is better at implicit reasoning even without the explicit SCM step.

– **The Completion Reversal (Level 4):** We observed a behavioral split at Level 4 (Partial VGDL). **Qwen3-8B** reverted to Direct Synthesis, which won **62.5%** of votes and achieved a higher Code Similarity (0.955 vs 0.939). This confirms that for smaller models, the overhead of generating a full SCM for simple completion tasks may introduce noise. Conversely, **QwQ-32B** maintained a preference for SCM (51.5% wins) and achieved higher Code Similarity (0.909 vs 0.876) with the causal graph. This suggests that reasoning-optimized models can effectively leverage the SCM structure even when the task is primarily syntactic completion.

Furthermore, the SCM-guided VGDL generation was consistently preferred across the benchmark games, except *portals* in the case of QwQ-32B. A comprehensive breakdown of performance metrics across all games is provided in the appendix.

Table 2. Context Level Metrics. **NS** denotes Non-SCM (Direct Synthesis). Note the reversal at Level 4 for Qwen3-8B.

Model	Target	Non-SCM (NS)		SCM-Guided		Outcomes (Win %)		
		Code ↑	Sem ↑	Code ↑	Sem ↑	NS ↑	SCM ↑	Tie
Qwen3-8B	L0: Obs Only	0.563	0.736	**0.653**	**0.740**	8.3%	**91.7%**	0.0%
Qwen3-8B	L1: Grammar	0.670	0.733	**0.710**	**0.743**	9.4%	**84.4%**	6.2%
Qwen3-8B	L2: Exact Game	0.692	0.725	**0.716**	**0.768**	37.5%	**59.4%**	3.1%
Qwen3-8B	L3: Distractors	**0.694**	0.714	0.693	**0.755**	25.0%	**75.0%**	0.0%
Qwen3-8B	L4: Partial	**0.955**	**0.895**	0.939	0.871	**62.5%**	37.5%	0.0%
QwQ-32B	L0: Obs Only	0.532	0.701	**0.591**	**0.732**	23.5%	**58.8%**	17.6%
QwQ-32B	L1: Grammar	**0.647**	0.731	0.646	**0.742**	32.4%	**52.9%**	14.7%
QwQ-32B	L2: Exact Game	0.660	**0.790**	**0.661**	0.760	11.8%	**82.4%**	5.9%
QwQ-32B	L3: Distractors	**0.659**	0.752	0.656	**0.764**	26.5%	**67.6%**	5.9%
QwQ-32B	L4: Partial	0.876	0.868	**0.909**	**0.883**	45.5%	**51.5%**	3.0%

Code: VGDL Cosine Similarity; **Sem**: Semantic Text Similarity.

5 Limitations, Future Work and Conclusion

Our semantic clustering relied on text embeddings that may not have fully captured the mechanical nuances of the games, potentially resulting in overlapping clusters; future work should evaluate alternative embedding architectures and clustering algorithms. Computational constraints (maximum 48 GB VRAM) restricted our analysis to models with $\leq$ 32B parameters and limited the synthesis task to a single observation set per game.

Future work should validate the SCM framework by human experts and evaluate its efficacy on a broader diversity of state-of-the-art architectures, such as

Llama, Mistral, or other closed-source models, and explore its applicability to other established AI testbeds like Stratega [22] and Tabletop games [9]. Current evaluation metrics for VGDL generation, such as simple parsability checks, are often arbitrary and do not adequately measure causal induction. Developing robust metrics for causal fidelity is a critical next step. A "belief SCM" framework where an LLM iteratively updates an internal causal model as it processes observations may be explored in the future. This approach would mitigate the high token overhead of batch processing while mimicking continuous learning.

These intermediate SCMs would also enable Causal Reinforcement Learning [5], facilitating interpretable agents. Perturbing these graphs also enhances Procedural Content Generation (PCG) by ensuring mechanical consistency, which addresses the playability limitations of statistical approaches [18] and allows for RL agent training through curriculum learning [13].

In conclusion, we demonstrate that grounding Large Language Models in Structural Causal Models significantly improves the reverse-engineering of game mechanics from raw observations. Across diverse games, context regimes, and models, the SCM-guided pipeline consistently yields VGDL rules that are more accurate and logically consistent than direct synthesis alone, particularly in low-information and inductive settings. These results provide empirical evidence that explicitly structured causal reasoning offers a powerful inductive bias beyond purely statistical pattern matching, establishing a foundation for neurosymbolic agents that can more robustly infer, explain, and manipulate the rules governing their environments.

Acknowledgments. This work is jointly supported by the "HybrInt - Hybrid Intelligence through Interpretable AI in Machine Perception and Interaction" project (Zukunft Nds, Niedersächsisches Ministerium für Wissenschaft, Grant ID: ZN4219). Parts of the work presented in this article are supported by Novo Nordisk Foundation grant NNF25OC0105856. The authors would like to thank Dr Diego Perez-Liebana for providing us with expert descriptions of the GVGAI games. The editing process was supported by Google's Gemini and experimentation was supported by ChatGPT.

References

1. Baier, C., Coenen, N., Finkbeiner, B., Funke, F., Jantsch, S., Siber, J.: Causality-based game solving. In: Silva, A., Leino, K.R.M. (eds.) International Conference on Computer Aided Verification, pp. 894–917. Springer (2021). https://doi.org/10.1007/978-3-030-81685-8_42
2. Bateni, B., Pratt, B., Whitehead, J.: LLM game rule understanding through out-of-distribution fine-tuning. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 21, pp. 2–11 (2025)
3. Bateni, B., Pratt, B., Whitehead, J.: Rule synergy analysis using LLMS: state of the art and implications. arXiv preprint arXiv:2508.19484 (2025)
4. Dave, N., Kifer, D., Giles, C.L., Mali, A.: Investigating symbolic capabilities of large language models. arXiv preprint arXiv:2405.13209 (2024)

5. Deng, Z., Jiang, J., Long, G., Zhang, C.: Causal reinforcement learning: a survey. arXiv preprint arXiv:2307.01452 (2023)
6. Dockhorn, A.: Prediction-based search for autonomous game-playing (2020). Ph.D. thesis
7. Dockhorn, A., Lucas, S.M., Volz, V., Bravi, I., Gaina, R.D., Perez-Liebana, D.: Learning local forward models on unforgiving games. arXiv preprint arXiv:1909.00442 (2019)
8. Dutta, S., Pandey, I., Singh, J., Manchanda, S., Chakrabarti, S., Chakraborty, T.: Frugal LMS trained to invoke symbolic solvers achieve parameter-efficient arithmetic reasoning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 17951–17959 (2024)
9. Gaina, R.D., Balla, M., Dockhorn, A., Montoliu, R., Perez-Liebana, D.: Design and implementation of tag: a tabletop games framework. arXiv preprint arXiv:2009.12065 (2020)
10. Hammond, K.J., Leake, D.B.: Large language models need symbolic AI. In: NeSy, pp, 204–209 (2023)
11. Hammond, L., Fox, J., Everitt, T., Carey, R., Abate, A., Wooldridge, M.: Reasoning about causality in games. Artif. Intell. **320**, 103919 (2023)
12. Hu, C., Zhao, Y., Liu, J.: Game generation via large language models. In: 2024 IEEE Conference on Games (CoG), pp. 1–4. IEEE (2024)
13. Jiwatode, M., Schlecht, L., Dockhorn, A.: Online optimization of curriculum learning schedules using evolutionary optimization. In: 2024 IEEE Conference on Games (CoG), pp. 1–8. IEEE (2024)
14. Li, C., Flanigan, J.: Task contamination: language models may not be few-shot anymore. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 18471–18480 (2024)
15. Mathieu, M., et al: AlphaStar unplugged: large-scale offline reinforcement learning. arXiv preprint arXiv:2308.03526 (2023)
16. McInnes, L., Healy, J., Melville, J.: UMAP: uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426 (2018)
17. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
18. Nielsen, T.S., Barros, G.A., Togelius, J., Nelson, M.J.: Towards generating arcade game rules with VGDL. In: 2015 IEEE Conference on Computational Intelligence and Games (CIG), pp. 185–192. IEEE (2015)
19. Pan, L., Albalak, A., Wang, X., Wang, W.Y.: Logic-LM: empowering large language models with symbolic solvers for faithful logical reasoning. arXiv preprint arXiv:2305.12295 (2023)
20. Pearl, J.: Causality. Cambridge University Press (2009)
21. Pearson, K.: LIII on lines and planes of closest fit to systems of points in space. London, Edinburgh, Dublin Philosop. Mag. J. Sci. **2**(11), 559–572 (1901)
22. Perez-Liebana, D., Dockhorn, A., Grueso, J.H., Jeurissen, D.: The design of "stratega": A general strategy games framework. arXiv preprint arXiv:2009.05643 (2020)
23. Perez-Liebana, D., Lucas, S.M., Gaina, R.D., Togelius, J., Khalifa, A., Liu, J.: General Video Game Artificial Intelligence, vol. 3. Morgan & Claypool Publishers (2019). https://gaigresearch.github.io/gvgaibook/
24. Reimers, N., Gurevych, I.: Sentence-BERT: sentence embeddings using SIAMESE BERT-networks. arXiv preprint arXiv:1908.10084 (2019)
25. Schaul, T.: An extensible description language for video games. IEEE Trans. Comput. Intell. AI Games **6**(4), 325–331 (2014)

26. Schmöcker, R., Dockhorn, A., Rosenhahn, B.: Accelerating probabilistic planning research: high-speed implementations of stochastic Markov decision processes and their catalogization. IEEE Access **13**, 193430–193461 (2025)
27. Silver, D., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018)
28. Tanaka, T., Simo-Serra, E.: Grammar-based game description generation using large language models. IEEE Trans. Games (2024)
29. Tanaka, T., Simo-Serra, E.: Grammar and gameplay-aligned RL for game description generation with LLMS. arXiv preprint arXiv:2503.15783 (2025)
30. Todd, G., Padula, A.G., Stephenson, M., Piette, É., Soemers, D.J., Togelius, J.: Gavel: generating games via evolution and language models. Adv. Neural. Inf. Process. Syst. **37**, 110723–110745 (2024)
31. Wei, J., et al.: Chain-of-thought prompting elicits reasoning in large language models. Adv. Neural. Inf. Process. Syst. **35**, 24824–24837 (2022)