One POMDP, Many Stories: LLM-driven Framework for Equivalence Verification

Anonymous ACL submission

Abstract

Different formulations of sequential decision processes make decision-making under uncertainty inefficient. Revealing deeper knowledge about the structure of decision problems by identifying and verifying equivalence across diverse formulations such as in Partially Observable Markov Decision Processes (POMDPs) can help but cost significant expert interventions. We propose a bidirectional translation iteration approach using large language models (LLMs) to systematically verify the equivalence between POMDPs and their textual descriptions. Our approach involves (1) generating a text description from a mathematical model of POMDP, (2) reconstructing a mathematical model from the description, and (3) comparing the reconstructed and the original models. To further test robustness, we introduce noise elements in states, actions, and observations to assess whether our approach recognizes them as equivalent.73% of our POMDPs were perfectly reconstructed from textual descriptions, and 89% remained structurally equivalent even under noise, leading to two distinct but equivalent textual versions for 63% of all models. Most discrepancies manifested as false negatives, reflecting the inherent ambiguity of natural language. Iterative refinement proved effective in reducing errors, though rare cases required additional iterations or more accurate initial descriptions. These results demonstrate that we can model multiple descriptions implying the same POMDP model, and this complex POMDP can be relatively accurately reconstructed from text descriptions. It also proves that iteratively refining the LLM outputs can significantly decline omissions or inaccuracies.

1 Introduction

The Partially Observable Markov Decision Process (POMDP) is a theoretical framework for decision-making under uncertainty. It extends the traditional Markov Decision Process (MDP) by modeling scenarios where an agent cannot fully observe the underlying state (Clark-Turner and Amato, 2017; Kaelbling et al., 1998). Due to its ability to explicitly handle partial observability, POMDPs have been applied in autonomous robotics, manufacturing, medical decision-making, and industrial systems (Capitán et al., 2014; Ong et al., 2009; Regan et al., 2010; Golowich et al., 2022). However, defining and modifying these mathematical models from a large amount of text data can require substantial human effort and is time consuming (Capitán et al., 2013; Bai et al., 2015).

Large Language Models (LLMs) have gained attention for their potential to directly derive a mathematical structure from text or, conversely, to automatically explain a mathematical model in natural language (Fagbohun et al., 2024; Imani et al., 2023; Collins et al., 2024). Because LLMs process vast amounts of textual data, they can extract POMDP elements from a textual description of a problem or transform a predefined POMDP into an interpretable text (Chadès et al., 2021; Jadhav, 2023). This approach aligns with the historical progression of computational linguistics, which has continuously connected "human-friendly language" with "machine-interpretable structures", evolving from basic machine languages to high-level languages and modern natural language processing systems.

Despite these advancements, a persistent challenge remains: different textual expressions can potentially describe the same POMDP structure, yet subtle variations in phrasing can lead them to appear entirely distinct. It is also possible for clearly distinct structures to appear similar. This problem raises a fundamental question: "How can one verify whether a textual description and a mathematical model are equivalent?" Our study responds to this question by proposing an approach that begins with a formally established POMDP, (1) transforms it into a textual description through an LLM, (2) re-

constructs a structural representation (such as a dictionary or a graph) from the generated text, and (3) checks whether it matches the original model. If (3) results in a mismatch, we provide LLM the specifications of the issue, and iteratively refine the textual description. This iteration process enforces the resulting textual description to become fully equivalent to the POMDP. We also investigate whether adding 'noise' states, actions, or observations to language descriptions results in changes in the POMDP structure.

The primary research questions addressed in this study are as follows:

- RQ1: What processes and criteria are necessary to objectively and systematically determine whether different textual descriptions actually refer to the same POMDP?
- RQ2. When a predefined POMDP is translated into textual description and then converted back to a model, under what conditions is its equivalence preserved or broken?
- RQ3. How do 'noise elements', such as split or merged actions, influence the verification of the equivalence of a POMDP in that process?

The main contributions of this study are threefold. First, we present an iterative framework that systematically verifies the equivalence between language descriptions and a mathematical model. This framework begins with a POMDP model and uses LLM-based text generation and reconstruction to ultimately form highly accurate language descriptions. Second, by deliberately introducing noise elements into textual descriptions, we confirm that the underlying decision-making structure remains unchanged, showing that multiple text representations can indeed coexist for the same POMDP. When these varied descriptions are reintroduced into the LLM, they are all recognized as referring to the same model, demonstrating that redundant or non-critical details do not compromise the core decision logic. This finding provides a robust basis for managing textual complexities in real-world scenarios. Furthermore, the ability to generate synthetic variations of text-based descriptions from a single POMDP opens opportunities to enrich and broaden LLM-based modeling approaches across diverse applications. Third, by bridging the gap between language descriptions and mathematical models, we propose a new possibility for simultaneously achieving the rigor of mathematical models and the intuitive clarity of the textual description. This is of direct relevance to domains such as autonomous driving, medical decision making, and data mining, where explainable AI and automated decision processes are of growing importance.

2 Related Works

2.1 POMDP Modeling: Theoretical Background and Industrial Applications

A POMDP (Smallwood and Sondik, 1973) extends the MDP(Bellman, 1957, 1966) to handle noisy or incomplete observations(Littman, 2009b). Formally:

$$\mathcal{P} = (S, A, O, P, Z, R), \tag{1}$$

where S is the set of states, A the set of actions, O the set of observations, $P(s' \mid s, a)$ the transition probability, $Z(o \mid s', a)$ the observation probability, and $R(s, a)$ the reward function. The agent maintains a belief distribution over states rather than directly observing them, making POMDPs common in robotics and healthcare, where sensing is imperfect.

Techniques such as point-based value iteration and Monte Carlo-based methods have helped address the computational challenges of Partially Observable Markov Decision Processes (POMDPs) (Spaan and Vlassis, 2005; Pineau et al., 2006). These advances have enabled successful applications in areas like autonomous robotics, manufacturing, logistics, and healthcare (Littman, 2009a; Jaulmes et al., 2005; Dutech and Scherrer, 2013). However, continuously updating and refining POMDP models remains difficult in large-scale industrial contexts—where states or observations frequently change—because it requires both domain expertise and mathematical modeling skills (Kavaklioğlu, 2024; Kavaklioğlu and Çevik, 2021).

2.2 Textual Description and Large Language Model (LLM) Approaches to Decision Modeling

Recent works have explored the use of natural language processing (NLP) and large language models (LLMs) to simplify and speed up the modeling process (Arco et al., 2020). Traditional NLP-based approaches often rely on constrained vocabularies or text formats (Trivedi et al., 2019; Garrido-Merchán et al., 2023). By contrast, LLMs trained on general text can identify components such as states, transitions, and rewards as well as convert mathematical models into more understandable text (Wu

et al., 2022; Feichter and Schlippe, 2024). However, the variability of natural language leads to questions about whether different textual descriptions indeed represent the same underlying model (Sànchez-Ferreres et al., 2017; Hoffmann et al., 2009). Even slight wording differences can obscure identical transition functions and rewards, underscoring the need for robust equivalence checks (Aa et al., 2017).

2.3 Model Equivalence Verification: Methods and Limitations

Researchers have addressed MDP/POMDP equivalence verification through formal verification, semantic parsing, and graph-based comparisons (Gros et al., 2020; Chatterjee and Henzinger, 2011; Chatterjee et al., 2011). These approaches often focus on structural similarities of transitions and states. Iterative verification methods can further compare rules or rewards to confirm that models yield the same policy (Brázdil et al., 2014). Yet, existing methods falter when models are re-expressed through additional "noise states" or "noise actions," which do not alter the policy but change the representation (Bork et al., 2020; Norman et al., 2017).

2.4 Remaining Issues and Limitations in Related Research

Despite incremental progress, a fully automated system that (1) reconstructs a mathematical model from a textual description, (2) ensures its equivalence to an original POMDP, and (3) verifies that "noise" or contextual additions do not compromise the optimal policy is still lacking. While some studies propose alternating between mathematical models and its textual description with LLMs, they are usually limited to narrow domains. There is no established protocol to conclusively prove that a POMDP's core policy remains unaltered across varied linguistic descriptions or added noise elements.

In response, this study proposes a methodology that uses LLM-based text generation and iterative verification to determine model equivalence more precisely. Our goal is to empower both domain experts and modelers to view POMDPs through transparent textual descriptions and rigorous mathematical structures, ensuring consistency and accuracy in complex, real-world environments.

3 Methodology

In this section, we present a two-step methodology for converting POMDP files from the `pomdp.org` repository into textual descriptions, and back. We verify correctness at each stage, and then perturb noise elements that do not make changes to the equivalence with the original model.

Algorithm 1 CHECKPOMDPEQUIVALENCE(P_1, P_2, ϵ)

Input: Two POMDPs P_1 and P_2; tolerance ϵ (e.g., 10^{-9})

Output: Boolean indicating whether P_1 and P_2 are equivalent

Check discount factor:
if $|P_1.\text{discount} - P_2.\text{discount}| > \epsilon$ **then**
  return FALSE
end if
Compute canonical forms:
$C_1 \leftarrow \text{CANONICALFORM}(P_1, \epsilon)$
$C_2 \leftarrow \text{CANONICALFORM}(P_2, \epsilon)$
Compare main components:
if Any transition probability in C_1 differs from C_2 by more than ϵ **then**
  return FALSE
end if
if Any observation probability in C_1 differs from C_2 by more than ϵ **then**
  return FALSE
end if
if Any reward value in C_1 differs from C_2 by more than ϵ **then**
  return FALSE
end if
return TRUE

3.1 Stage A: POMDP Transformation and Reconstruction

Figure 1 illustrates the initial sequence used to transform a POMDP file into a verified text–dictionary pair. The six labeled steps in the figure are detailed below.

(1) Dataset Construction. POMDP files are collected from `pomdp.org`. Each file encodes a set of states, actions, observations, transition probabilities, observation probabilities, and rewards.

(2) Ground Truth Extraction. Each POMDP file is parsed to capture the essential elements: (i) lists of states, actions, and observations; (ii) a discount factor; and (iii) probabilities for transitions, observations, and rewards. This parsing step ensures that all probabilities and rewards are explicitly enumerated.

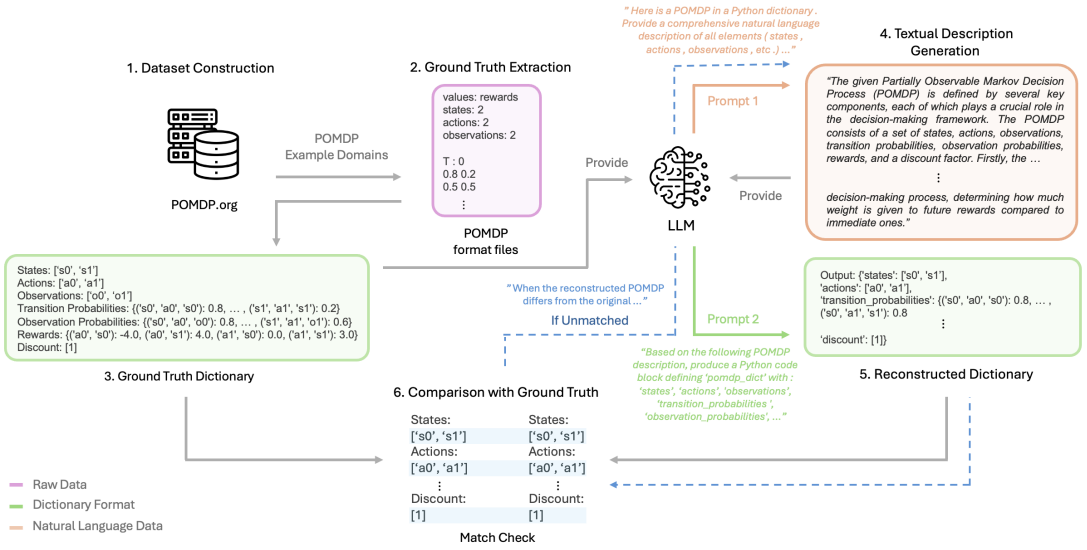

Figure 1: Overview of Stage A: POMDP Transformation and Reconstruction. The stage begins with collecting POMDP domains (Step 1), extracting ground truth (Step 2), creating a fully specified dictionary (Steps 2–3), generating its textual description (Step 4), and then reconstructing a POMDP element dictionary from that text for comparison against the ground truth(Steps 5–6).

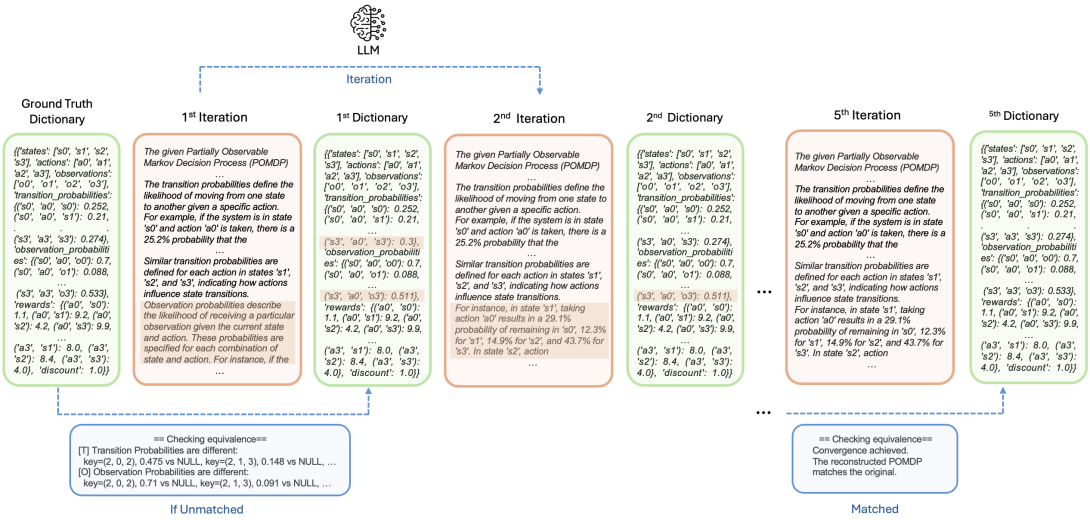

Figure 2: Overview of Refinement Iteration Process

(3) Ground Truth Dictionary. Once extracted, all POMDP is stored in a dictionary-based format. An illustrative example (inset, lower left of Figure 1) shows how states, actions, and other fields are recorded. This dictionary is considered the *ground truth* representation for the corresponding POMDP.

(4) Textual Description Generation. The ground truth dictionary is then provided to a Large Language Model (LLM)—here, GPT-4o—via a prompt that requests a complete, self-contained English description. This prompt explicitly requires listing every probability and reward. The generated output constitutes a human-readable text that captures the dictionary's content.

(5) Reconstructed Dictionary. A second prompt is sent to the same LLM, instructing it to reconstruct the POMDP dictionary from the textual description. This reconstructed dictionary is compared against the ground truth.

(6) Comparison with Ground Truth. The final step in Figure 1 compares the reconstructed dictionary (Step 5) to the original ground truth (Step 3). If they match under the equivalence criterion (Algorithm 1), the transformation is considered successful. If not, the process returns to Step 4, where a refined textual description is requested based on the feedback, and Steps 4–6 repeat until a match is

Algorithm 2 CANONICALFORM(P, ϵ)

Input: POMDP P with states S, actions A, observations O, transitions T, observation map O_b, rewards R, discount d; tolerance ϵ
Output: $(\widehat{T}, \widehat{O}_b, \widehat{R}, d)$: the canonical reduction of P

Remove unreachable states:
Let s_0 be the first state in S
Perform BFS from s_0 over T using edges with probability $> \epsilon$
Remove states not reached
Remap indices:
Reassign integer indices to remaining S, A, O
Rebuild T, O_b, R accordingly
Iterative pruning:
repeat
    Remove any s, a, or o not appearing in T, O_b, or R above threshold ϵ
    Remove noise actions (e.g., self-loops with zero reward and uniform observations)
until no further elements removed
Return canonical form:
return $(\widehat{T}, \widehat{O}_b, \widehat{R}, d)$

achieved or a predefined iteration limit is reached. Figure 1 illustrates the refinement process starting from original POMDP to equivalent textual description. Also, Figure 2 illustrates holistic overview of iteration refinement process.

Two POMDP dictionaries are considered *equivalent* if they share the same discount factor (to within a small tolerance ϵ) and, after unreachable states and noise elements are pruned, they exhibit the same transition, observation, and reward mappings. Algorithm 1 takes as input two POMDPs, P_1 and P_2, and a tolerance parameter ϵ. Each POMDP is represented by its standard components:

$$P_i = (S_i, A_i, O_i, T_i, O_{b,i}, R_i, d_i),$$

where S_i is the set of states, A_i the set of actions, O_i the set of observations, T_i the transition probabilities, $O_{b,i}$ the observation probabilities, R_i the rewards, and d_i the discount factor. Algorithm 2 then transforms each P_i into its canonical form by removing (i) any states that are unreachable from a designated start state and (ii) any noise actions that are pure self-loops with zero reward. After these reductions, the canonical forms of P_1 and P_2 (denoted C_1 and C_2) are checked in Algorithm 1 to ensure that all transition probabilities, observation probabilities, and rewards match within ϵ. If they do, the two POMDPs are deemed equivalent. Otherwise, the check fails.

3.2 Stage B: Noise Perturbation and Reverification

Once a minimal text–dictionary pair has been verified as equivalent, noise elements are introduced into the textual description to ascertain whether the reconstruction process still yields a POMDP identical to the original dictionary. Figure 3 illustrates this second stage, again enumerated in six steps.

(1) Textual description & (2) Noise perturbation. A previously verified text (from Stage A) is provided to a different LLM (Claude-3.5-sonnet), which adds extra states, actions, or observations. These new elements are *inconsequential* in that they do not alter transition probabilities or rewards for the original subset of states and actions. They may, for example, be self-loops with zero reward and uniform observation distributions. *A separate LLM is intentionally used at this stage to lessen the risk that the same model (GPT-4o) recognizes and discards or modifies these artificial elements, thereby reducing the likelihood of unintentionally invalidating the noise perturbation process.*

(3) Pre-Extracted Dictionary & (4) Reconstructed Dictionary with Noise Elements. The perturbed text is submitted to the reconstruction process (similar to Steps 4–5 in Figure 1), yielding a new dictionary that includes both original and noise components.

(5) Comparison. This new dictionary is compared with the original ground truth dictionary using Algorithm 1. The check prunes unreachable and noise elements, so it should return TRUE if the core model remains consistent.

(6) Match Check and Iteration. If the match succeeds, the noise-enhanced text is accepted as a valid alternative representation. Otherwise, we revise the perturbation prompt, and Steps 1–5 of Figure 3 and repeat until convergence or until a maximum iteration limit is reached.

We employ three core prompts that respectively (1) convert a Python dictionary to a textual description, (2) reconstruct the dictionary from a textual description, and (3) perturb the text with noise elements. Figure 14 to Figure 16 illustrate these prompt examples used in our methodology. Figure 17 shows the prompt for refinement process when the generated text does not match with the original POMDP.

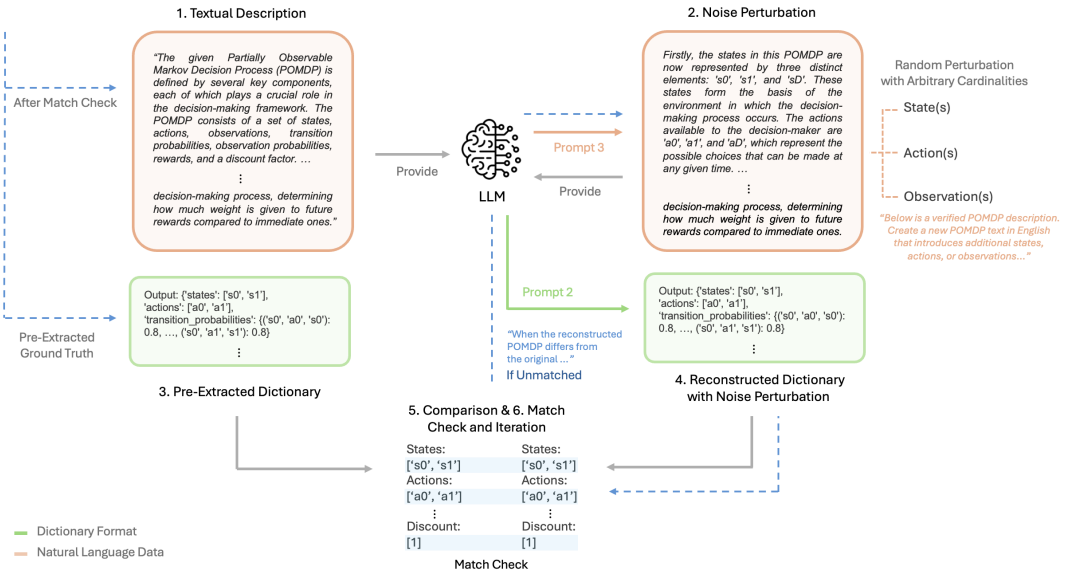

Figure 3: Overview of Stage B: Noise-Element Perturbation and Reverification. A previously validated natural-language text is perturbed with new noise states, actions, or observations (Steps 1–2). The resulting text is reconstructed into a dictionary (Steps 3–4) and checked for equivalence (Steps 5–6).

4 Results

4.1 Equivalent Textual descriptions

We found that 73% of the POMDPs in our dataset admit textual descriptions that, upon reconstruction, match the original models exactly. Within that subset of verified POMDPs, 89% can be further perturbed with additional (noise) states, actions, or observations while preserving the same underlying decision-making structure. Consequently, 63% of all POMDPs end up with *two distinct textual descriptions*—the original verified version and its noise-perturbed counterpart—both recognized as equivalent to the original.

4.2 Error Pattern Analysis

We analyzed the error patterns in the extracted POMDPs by focusing on the frequency and nature of deviations in the LLM outputs, rather than relying solely on an aggregated per-problem accuracy metric. Although many studies report the precision of such modeling as a percentage, we avoided this approach for two reasons. First, errors in one component (e.g., `states`) can propagate to related elements such as `transition probabilities` (T) and `observation probabilities` (O), making separate assessments of each element potentially misleading. Second, matrix-based comparisons often overemphasize minor numerical discrepancies while underrepresenting serious structural omissions, as both are typically counted as a single error.

To further investigate these deviations, we examined the specific errors occurring at each iteration of each problem. Notably, we observed no errors in the `states`, `actions`, or `observations` dimensions. Instead, all discrepancies arose in T, O, and `rewards` (R), as summarized in Table 1. We classified errors according to their alignment with the ground truth: a description was considered a true positive only if it contained all relevant `state`, `action`, and `observation` combinations and matched their exact values. Thus, in addition to conventional false positives and false negatives, we also treated mismatched values within correctly identified structures as errors. We use standard classification terms as follows: False Positive (FP) means the ground truth is zero(or non-existant) but the generated text is non-zero, False Negative (FN) means the ground truth is non-zero but the generated text is zero(or non- existant), and True Positive (TP) mismatch means both are non-zero but differ. T, O, and R refer to transition-probability, observation-probability, and reward errors, respectively. As shown in Table 1, most errors were false negatives, where ground truth values existed but were not fully captured by the textual description. This outcome suggests that the inherent ambiguities of natural language can hinder the precise representation of POMDP structures.

Table 1: Global error pattern summary across all problems.

Error Type	Total	False Positive (FP)	False Negative (FN)	TP mismatch
T	3448	57 (1.7%)	3359 (97.4%)	32 (0.9%)
O	4050	39 (1.0%)	4011 (99.0%)	0 (0.0%)
R	831	0 (0.0%)	831 (100.0%)	0 (0.0%)

Table 2: Iterative Refinement Results

Trend	Transition	Observation	Reward
Error Increase	0%	3%	3%
Error Decrease to Constant	22%	22%	14%
Error Decrease to Zero	11%	8%	3%
Success at First Iteration	61%	64%	77%
No Changes in Error	6%	3%	3%

4.3 Iterative Refinement Results

We employed iterative refinement as a mechanism to mitigate the uncertainty in LLM-generated outputs and to emphasize the importance of prompt design. A common criticism of LLM-based research is the inconsistency of generated results. However, our findings demonstrate that consistent outputs can be achieved through an iterative approach.

Figure 4 illustrates the accuracy trends over multiple iterations. In Figures 4 (a) and (b), errors decrease with each iteration, ultimately resulting in an equivalent representation with the ground truth. Specifically, in Figure 4 (a), equivalence was achieved after two iterations, whereas in Figure 4 (b), three iterations were required. Figures 4 (c) and (d) exhibit a general trend of decreasing errors as the number of iterations increases. In such cases, we can expect that increasing the maximum number of iterations would result in convergence to the ground truth. Meanwhile, case in which errors increased with iterations was observed only once across the entire dataset. Additionally, certain cases exhibited persistent errors that did not improve with further iterations. These cases can be attributed to either an insufficient number of iterations or severe defects in the initial textual description, rendering refinement ineffective.

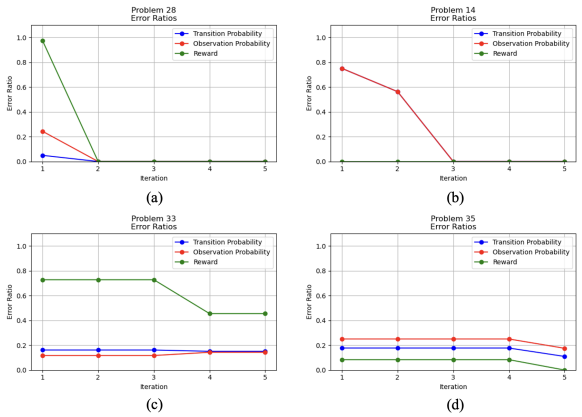

Figure 4: Iterative Refinement Results: (a), (b) indicate decreasing Error and convergence, while (c), (d) indicate decreasing error trend while they do not fully converge

Moreover, in this experiment, the textual descriptions used were relatively simple, leading to direct reconstruction in approximately 60–70% of cases during the first iteration. However, in real-world applications, more complex textual inputs are expected, suggesting that the potential of iterative refinement is substantial.

5 Conclusion

In this paper, we present a bidirectional, iterative framework for verifying equivalence between textual descriptions and their underlying POMDP representations, motivated by the need to reconcile human-friendly language with rigorous decision-theoretic models under partial observability. Our approach starts with a ground-truth POMDP, generates a textual narrative describing it, and then reconstructs the POMDP from that description using an LLM. By comparing the reconstructed model against the original, we iteratively refine any inconsistencies until an equivalent textual description is obtained. We also introduce artificial "noise elements" (additional states, actions, or observations) to test whether the core structure remains intact, revealing that multiple narratives can faithfully encode the same decision model. The primary challenge lies in accurately capturing transitions, observations, and rewards, but an iterative refinement process addresses these issues, ultimately bridging intuitive language descriptions with rigorous probabilistic models.

6 Future Work

Several avenues for extending this research include systematically addressing scenarios in which a single action is represented by multiple sub-actions in the text (or vice versa) while retaining policy equivalence, exploring more complex transformations such as partial merges of observations or domain-specific synonyms to verify structural consistency, examining scalable verification mechanisms—potentially approximate or sampling-based—to handle the high-dimensional or continuous state spaces encountered in real-

world applications, and conducting broader comparisons across various large language models and decision-making domains to evaluate the robustness of textual–model equivalence and guide future improvements in prompt engineering and model selection.

Overall, our findings highlight how an iterative, LLM-based approach can effectively bridge the gap between natural language and mathematical representations of decision processes under uncertainty. By refining textual narratives until they align with a formal POMDP, we offer a pathway toward enhancing explainability and manageability in complex, partially observable decision systems. We hope that this work will serve as a foundation for future research on more flexible, scalable, and robust automated model-building pipelines, ultimately advancing decision-making in real-world settings.

7 Limitations

Limited Noise Scope. This research primarily centered on introducing self-contained noise elements (i.e., additional states, actions, or observations that do not affect reward outcomes). More complex perturbations—such as splitting a single action into multiple sub-actions or merging multiple states—were not systematically examined, although such transformations can similarly preserve policy equivalence.

Prompt Engineering. Ensuring accurate convergence frequently entails multiple iterative refinements and the incorporation of manual feedback on mismatched components. Achieving a fully automated iteration and feedback mechanism in a robust, domain-agnostic manner remains a non-trivial challenge. Although we tested through multiple prompting methods including three-shot, Chain-of-Thought, and Zero-shot, there may be additional choices for optimal prompting.

References

H. v. d. Aa, H. Leopold, and H. A. Reijers. 2017. Comparing textual descriptions to process models – the automatic detection of inconsistencies. *Information Systems*, 64:447–460.

L. Arco, G. Nápoles, F. Vanhoenshoven, A. L. Lara, G. Casas, and K. Vanhoof. 2020. Natural language techniques supporting decision modelers. *Data Mining and Knowledge Discovery*, 35:290–320.

H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee. 2015. Intention-aware online pomdp planning for autonomous driving in a crowd. *2015 IEEE International Conference on Robotics and Automation (ICRA)*.

Richard Bellman. 1957. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684.

Richard Bellman. 1966. Dynamic programming. *science*, 153(3731):34–37.

A. Bork, S. Junges, J. Katoen, and T. Quatmann. 2020. Verification of indefinite-horizon pomdps.

T. Brázdil, K. Chatterjee, M. Chmelík, V. Forejt, J. Křetínský, M. Kwiatkowska, D. Parker, and M. Ujma. 2014. Verification of markov decision processes using learning algorithms. *Lecture Notes in Computer Science*, pages 98–114.

J. Capitán, L. Merino, and A. Ollero. 2014. Decentralized cooperation of multiple uas for multi-target surveillance under uncertainties. *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*.

J. Capitán, M. T. J. Spaan, L. Merino, and A. Ollero. 2013. Decentralized multi-robot cooperation with auctioned pomdps. *The International Journal of Robotics Research*, 32:650–671.

I. Chadès, L. V. Pascal, S. Nicol, C. S. Fletcher, and J. Ferrer-Mestres. 2021. A primer on partially observable markov decision processes (pomdps). *Methods in Ecology and Evolution*, 12:2058–2072.

K. Chatterjee and M. Henzinger. 2011. Faster and dynamic algorithms for maximal end-component decomposition and related graph problems in probabilistic verification. *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1318–1336.

K. Chatterjee, M. Henzinger, M. Joglekar, and N. Shah. 2011. Symbolic algorithms for qualitative analysis of markov decision processes with büchi objectives. *Computer Aided Verification*, pages 260–276.

M. Clark-Turner and C. Amato. 2017. Cog-dice: an algorithm for solving continuous-observation decpomdps. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4573–4579.

K. M. Collins, A. Q. Jiang, S. Frieder, L. Wong, M. Zilka, U. Bhatt, T. Lukasiewicz, Y. Wu, J. B. Tenenbaum, W. Hart, W. T. Gowers, W. Li, A. Weller, and M. Jamnik. 2024. Evaluating language models for mathematics through interactions. *Proceedings of the National Academy of Sciences*, 121.

A. Dutech and B. Scherrer. 2013. Partially observable markov decision processes. *Markov Decision Processes in Artificial Intelligence*, pages 185–228.

O. Fagbohun, N. P. Iduwe, M. Abdullahi, A. Ifaturoti, and O. M. Nwanna. 2024. Beyond traditional assessment: exploring the impact of large language models on grading practices. *Journal of Artificial Intelligence, Machine Learning and Data Science*, 2:1–8.

C. Feichter and T. Schlippe. 2024. Investigating models for the transcription of mathematical formulas in images. *Applied Sciences*, 14:1140.

E. C. Garrido-Merchán, R. Gozalo-Brizuela, and S. González-Carvajal. 2023. Comparing bert against traditional machine learning models in text classification. *Journal of Computational and Cognitive Engineering*, 2:352–356.

N. Golowich, A. Moitra, and D. Rohatgi. 2022. Learning in observable pomdps, without computationally intractable oracles.

T. P. Gros, H. Hermanns, J. Hoffmann, M. Klauck, and M. Steinmetz. 2020. Deep statistical model checking. *Lecture Notes in Computer Science*, pages 96–114.

V. Hoffmann, H. Lichter, A. Nyßen, and A. Walter. 2009. Towards the integration of uml- and textual use case modeling. *The Journal of Object Technology*, 8:85.

S. Imani, L. Du, and H. Shrivastava. 2023. Mathprompter: mathematical reasoning using large language models.

E. a. V. D. Jadhav. 2023. Understanding the order of 500 and 1000 rupees notes ban using reinforcement learning. *International Journal on Recent and Innovation Trends in Computing and Communication*, 11:2482–2488.

R. Jaulmes, J. Pineau, and D. Precup. 2005. Active learning in partially observable markov decision processes. *Machine Learning: ECML 2005*, pages 601–608.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134.

C. Kavaklioğlu. 2024. Novel solution methods for cpomdps with applications to cancer screening.

C. Kavaklioğlu and M. Çevik. 2021. Scalable grid-based approximation algorithms for partially observable markov decision processes. *Concurrency and Computation: Practice and Experience*, 34.

M. L. Littman. 2009a. A tutorial on partially observable markov decision processes. *Journal of Mathematical Psychology*, 53:119–125.

Michael L Littman. 2009b. A tutorial on partially observable markov decision processes. *Journal of Mathematical Psychology*, 53(3):119–125.

G. Norman, D. Parker, and X. Zou. 2017. Verification and control of partially observable probabilistic systems. *Real-Time Systems*, 53:354–402.

S. Ong, S. Png, D. Hsu, and W. S. Lee. 2009. Pomdps for robotic tasks with mixed observability. *Robotics: Science and Systems V*.

J. Pineau, G. Gordon, and S. Thrun. 2006. Anytime point-based approximations for large pomdps. *Journal of Artificial Intelligence Research*, 27:335–380.

T. J. Regan, I. Chadès, and H. P. Possingham. 2010. Optimally managing under imperfect detection: a method for plant invasions. *Journal of Applied Ecology*, 48:76–85.

Richard D Smallwood and Edward J Sondik. 1973. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088.

M. T. J. Spaan and N. Vlassis. 2005. Perseus: randomized point-based value iteration for pomdps. *Journal of Artificial Intelligence Research*, 24:195–220.

J. Sànchez-Ferreres, J. Carmona, and L. Padró. 2017. Aligning textual and graphical descriptions of processes through ilp techniques. *Lecture Notes in Computer Science*, pages 413–427.

G. Trivedi, E. R. Dadashzadeh, R. Handzel, W. W. Chapman, S. Visweswaran, and H. Hochheiser. 2019. Interactive nlp in clinical care: identifying incidental findings in radiology reports. *Applied Clinical Informatics*, 10:655–669.

Y. Wu, A. Q. Jiang, W. Li, M. N. Rabe, C. Staats, M. Jamnik, and C. Szegedy. 2022. Autoformalization with large language models.

A Appendices

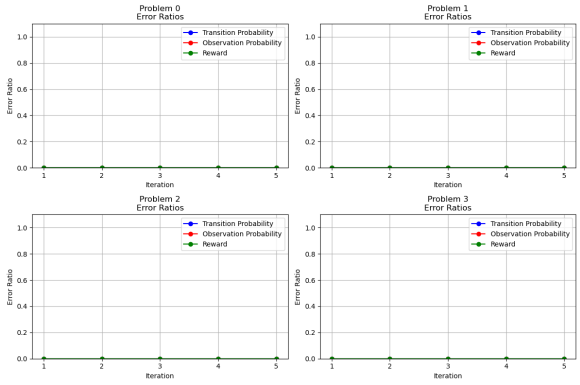

Figure 5: Problems 0-3

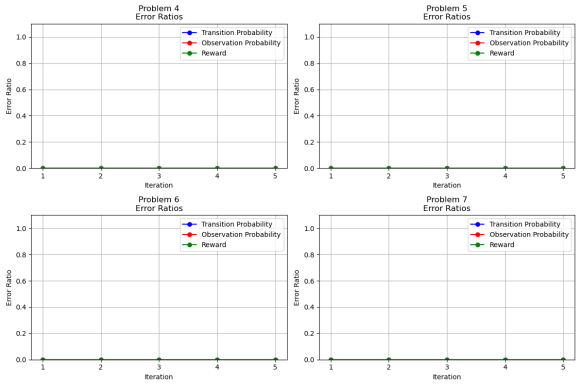

Figure 6: Problems 4-7

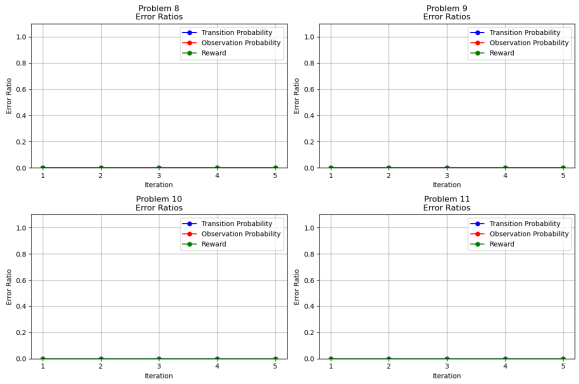

Figure 7: Problems 8-11

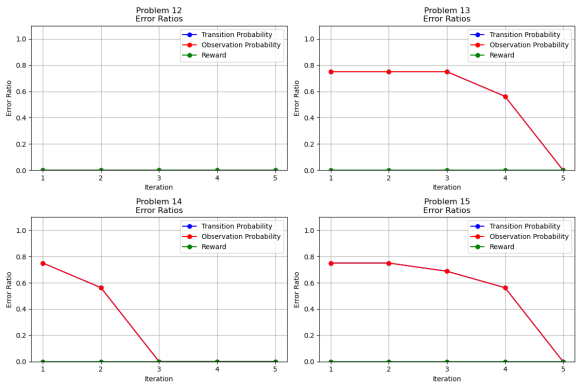

Figure 8: Problems 12-15

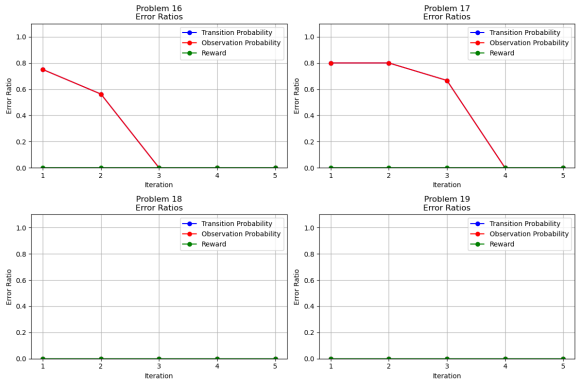

Figure 9: Problems 16-19

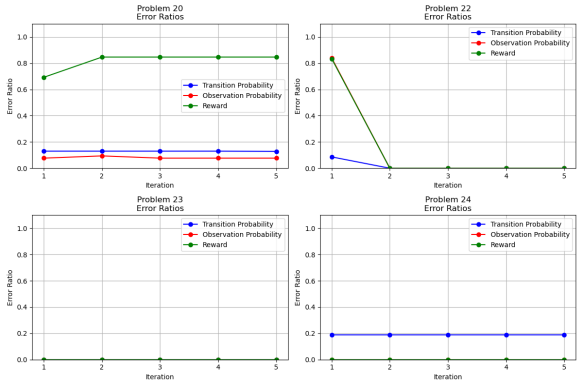

Figure 10: Problems 20-23

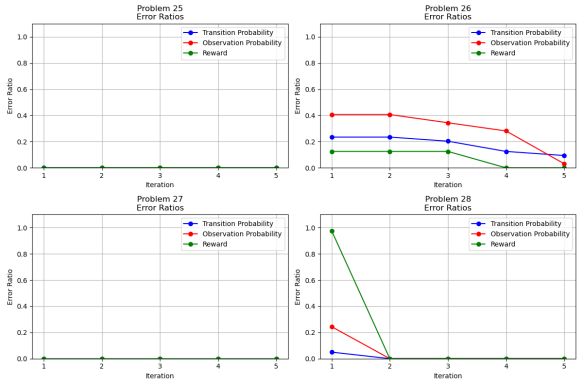

Figure 11: Problems 24-27

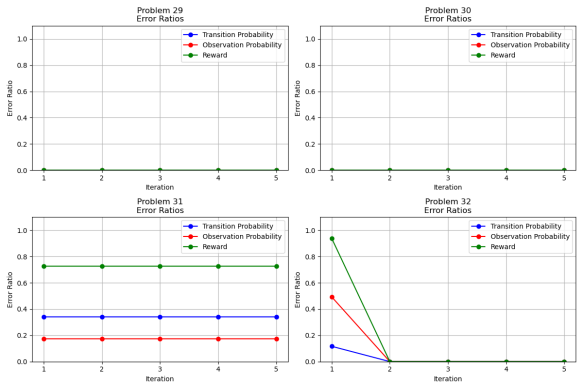

Figure 12: Problems 28-31

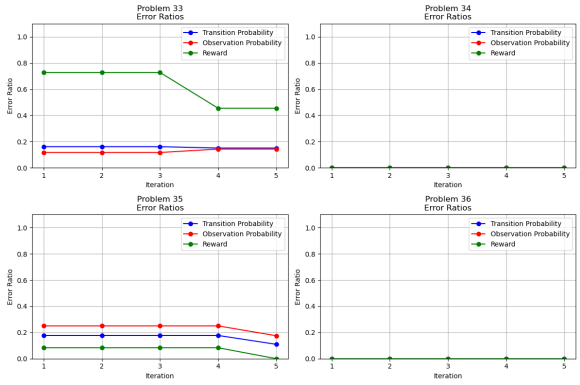

Figure 13: Problems 32-35

```
Prompt 1: Dictionary to Textual Description
-------------------------------------------
Example 1:
Example 2:
Example 3:

--- Now for your request ---
User:
"Here is a POMDP in a Python dictionary.
Provide a complete textual description of
all elements (states, actions, observations, etc.),
with nothing omitted.

Dictionary:
{<GROUND TRUTH DICTIONARY>}
"
```

Figure 14: Prompt 1: Converts a Python dictionary representation of a POMDP into a natural language description covering all components.

```
Prompt 2: Textual Description to Dictionary
-------------------------------------------
User:
"Based on the following POMDP description,
produce a Python code block defining 'pomdp_dict'
with 'states', 'actions', 'observations',
'transition_probabilities',
'observation_probabilities',
'rewards', and 'discount'.
Never omit any component.

Example format:
'states': [...],
'actions': [...],
'observations': [...],
'transition_probabilities': {("s0","a0","s0"): 0.8,
     ...},
'observation_probabilities': {("s0","a0","o0"): 0.7,
      ...},
'rewards': {("a0","s0"): 1.0, ...},
'discount': 1.0

Here is the POMDP description:
<INSERT NATURAL LANGUAGE TEXT>
"
```

Figure 15: Prompt 2: Translates a natural language POMDP description into a complete Python dictionary format.

```
Prompt 3: Noise Element Perturbation
------------------------------------
User:
"Below is a verified POMDP description.
Create a new POMDP text in English that
introduces additional states, actions,
or observations (e.g., self-loops) without
labeling them as 'noise' or irrelevant.
Include transitions, rewards, and discount
normally, with no mention of extra elements.

Original POMDP description:
<INSERT VERIFIED NATURAL LANGUAGE TEXT>
"
```

Figure 16: Prompt 3: Perturbs a valid POMDP description by adding extra elements while preserving coherence.

```
Prompt 4: Non-Convergence Refinement
------------------------------------
User:
"Below is the POMDP text you previously produced
     from
{<GROUND TRUTH DICTIONARY>}.
Unfortunately, it does not match the original
POMDP because of the following issues:

[CORRECTIONS DETAILS HERE]

Please incorporate these corrections into the
textual POMDP description so that it aligns
fully with the original. Simply ensure the refined
     text is
consistent with all required transitions,
     observations,
rewards, and discount factor, matching the original
     POMDP's
structure and values."
```

Figure 17: Prompt 4: Requests a refined textual description when equivalence fails, instructing the model to integrate corrections.