Bridging Probabilistic Graphical Models and Large Language Models via Structured Prompt Translation for Enhanced Reasoning and Uncertainty Awareness

Shafik Houeidi
Department of Electrical and Computer Engineering
American University of Beirut
Beirut, Lebanon
sah89@mail.aub.edu

Razan Tajeddine
Department of Electrical and Computer Engineering
American University of Beirut
Beirut, Lebanon
rt37@aub.edu.lb

Abstract

Large Language Models provide flexible language understanding but lack explicit mechanisms for calibrated uncertainty and structured dependence modeling. Probabilistic Graphical Models offer principled representations for causality and uncertainty with clear semantics yet have limited natural language interfaces. We present a modular interface that connects the two through a structured prompt translation layer. An encoder LLM compiles a natural language query into a minimal JSON schema for a Bayesian network, a validator enforces graphical and probabilistic constraints, a standard inference engine computes posteriors, and a decoder LLM verbalizes the outcome with uncertainty-aware explanations. The approach is training free and emphasizes separation of concerns. We specify design choices, schemas, validation rules, and an evaluation plan without reporting empirical results.

what two? be clear.

→ very long sentence.

→ why?

Write out acronym on first occurence (in abstract)

1 Introduction

Large Language Models (LLMs) such as GPT-4 and Claude have achieved remarkable success in natural language understanding and reasoning tasks, including question answering, summarization, and scientific explanation. Despite this progress, LLMs remain limited in their ability to perform *structured reasoning* and *quantify uncertainty*. Their outputs, while linguistically coherent, often lack internal consistency and probabilistic grounding. These limitations are particularly evident in high-stakes or knowledge-intensive domains, where decisions require explicit reasoning about dependencies and confidence levels [3, 6].

what does that mean?

Probabilistic Graphical Models (PGMs) provide a formal framework for representing and reasoning about uncertainty through graphs that encode conditional dependencies among random variables [3]. By separating model structure from parameters, PGMs enable transparent and interpretable reasoning, offering a principled approach to handle uncertainty and causal inference? Classical PGMs, such as Bayesian Networks and Markov Random Fields, have demonstrated strong capabilities in diagnostic reasoning, decision support, and scientific modeling. However, their applicability in open-ended, language-based environments is limited because constructing and interpreting PGMs from natural language remains non-trivial.

here can use PGMs after writing it in abstract.

things are not very clear here

Do not capitalize the acronym definition

Recent work has explored combining the probabilistic rigor of PGMs with the flexibility of language models. One line of research treats LLMs as stochastic components within probabilistic inference frameworks, using techniques such as Sequential Monte Carlo (SMC) [11, 4] and Approximate Bayesian Computation (ABC) [8] to sample from implicit posteriors. These approaches formalize uncertainty estimation but often rely on complex inference procedures and high computational cost. Another emerging direction focuses on *probabilistic prompting*, where LLMs are guided to reason about probabilities directly through structured textual templates or verbalized Bayesian dependencies. For example, Verbalized Probabilistic Graphical Modeling (vPGM) demonstrates that LLMs can express relationships such as $P(A|B)$ or $P(B|A, C)$ in natural language [2], achieving interpretable reasoning without explicit inference code. However, purely verbalized approaches remain limited by linguistic ambiguity and the absence of formal guarantees.

why bold?

In this work, we propose a modular framework that unifies these perspectives by integrating LLMs and PGMs through a structured *encoding–inference–decoding* pipeline. The framework introduces a lightweight **PGM interface layer** that translates natural language queries into formal probabilistic representations, performs rigorous inference, and returns interpretable results in natural language. Specifically, a *PGM encoder* reformulates user queries into structured probabilistic models defined by a JSON schema specifying variables, dependencies, and conditional probabilities. A dedicated *probabilistic inference engine*—implemented using open-source tools such as `pgmpy`, `PyMC` or `Pyro`—executes Bayesian inference to compute posterior distributions and credible intervals. Finally, a *PGM decoder* verbalizes the results, providing uncertainty estimates, causal insights, and counterfactual explanations.

how or? you are discussing your work

Unlike fine-tuned or specialized probabilistic LLMs, the proposed system requires no model retraining. Instead, it leverages grammar-constrained decoding, schema validation, and external inference modules to achieve consistency and transparency. This design separates the semantic understanding capabilities of LLMs from the mathematical rigor of probabilistic reasoning, addressing both interpretability and reliability.

why is this bold?

We demonstrate this architecture through a case study in **medical diagnosis under uncertainty**, where symptoms and diseases naturally form probabilistic dependencies. (For instance, the likelihood of pneumonia given observed symptoms such as fever and cough can be ~~modeled as $P(\text{Pneumonia}|\text{Fever, Cough})$,~~ computed through Bayesian inference, and explained to the user in natural language.) Our experiments compare this approach with standard Chain-of-Thought prompting and verbalized probabilistic reasoning, showing improvements in calibration, logical consistency, and interpretability.

The main contributions of this paper are:

1. A modular framework that integrates LLMs with probabilistic inference through structured encoding, inference, and decoding.
2. A training-free implementation using existing Bayesian inference engines for reliable uncertainty quantification.
3. A demonstration in medical diagnostic reasoning, showing improved interpretability and calibration compared with standard LLM prompting.
4. A reproducible prototype that bridges natural language understanding with formal probabilistic reasoning.

By grounding LLM reasoning in probabilistic structure, this work moves toward transparent, uncertainty-aware AI systems capable of verifiable and interpretable decision-making.

2 Related Work

2.1 Probabilistic Reasoning and Uncertainty in Language Models

You are redefining this, why?

While large language models (LLMs) demonstrate remarkable proficiency in linguistic understanding, they struggle to perform explicit probabilistic reasoning or represent uncertainty. Conventional prompting methods such as Chain-of-Thought (CoT) improve logical consistency but still lack calibrated uncertainty estimates and formal interpretability. Recent works have therefore explored embedding LLMs into probabilistic frameworks to bridge this gap. reference needed here.

Several approaches reinterpret the outputs of LLMs as stochastic samples, enabling approximate probabilistic inference. Sharma et al. [8] proposed the use of ~~Approximate Bayesian Computation~~ ~~(~~ABC~~)~~ for *uncertainty quantification* in LLMs, treating the model as a stochastic simulator that produces samples consistent with a posterior distribution. Similarly, Nafar et al. [6] introduced methods for reasoning over uncertain text, transforming unstructured language into structured Bayesian representations to quantify belief propagation across text-derived variables. These methods highlight that uncertainty can be elicited from LLMs, but they remain limited by implicit sampling processes rather than explicit graphical reasoning.

already defined.

2.2 Probabilistic Program Steering and Sequential Monte Carlo Approaches

Another line of research uses formal inference algorithms to *steer* or constrain language model behavior through probabilistic programs. Zhou et al. [11] introduced a framework that integrates ~~Sequential Monte Carlo~~ ~~(~~SMC~~)~~ algorithms with probabilistic programming to guide LLMs toward samples consistent with a target distribution. Luo and Mnih [4] further proposed *Twisted SMC*, which improves posterior sampling by introducing adaptive resampling and control variates, allowing language models to represent structured uncertainty more efficiently. These works demonstrate that probabilistic inference can improve sample efficiency and calibration, but they require significant computational overhead and tightly coupled probabilistic programming environments.

Our framework differs by using SMC and Bayesian inference not as direct steering mechanisms, but as modular inference engines external to the language model. This design allows the LLM to serve as a semantic translator, while dedicated inference libraries (e.g., PyMC or pgmpy) perform the mathematical reasoning. The resulting separation enhances modularity and reproducibility without necessitating specialized probabilistic programming environments.

2.3 Verbalized Probabilistic Reasoning and Symbolic Control

A complementary direction involves the *verbalization* of probabilistic reasoning. Huang et al. [2] introduced ~~*Verbalized Probabilistic Graphical Modeling*~~ ~~(~~vPGM~~)~~, in which LLMs express Bayesian dependencies directly in natural language, such as “the probability of pneumonia increases if both fever and cough are present.” This technique allows models to describe uncertainty qualitatively, yielding more interpretable responses. However, because vPGM relies solely on verbalized representations, it lacks formal computational guarantees and cannot produce quantitative posterior distributions or uncertainty intervals.

Our proposed encoder–inference–decoder framework can be viewed as a structured generalization of vPGM. Instead of relying on purely linguistic probability descriptions, we use the LLM to construct an explicit ~~probabilistic graphical model~~ ~~(~~PGM~~)~~ that encodes dependencies, priors, and conditionals in a machine-interpretable form. The inference step is then handled by a probabilistic engine that performs exact or approximate Bayesian inference, while the decoder translates the results back into interpretable natural language explanations.

2.4 Self-Steering and Meta-Reasoning Architectures

Recent work has also examined ways for LLMs to *self-steer* or recursively guide their own reasoning. Grand et al. [1] proposed the *Self-Steering Language Model* architecture, where the model acts as both planner and executor, generating its own reasoning trajectories to reach probabilistically consistent conclusions. These architectures represent a move toward autonomous, meta-cognitive systems capable of internal reasoning control. However, they remain vulnerable to hallucinations and lack explicit uncertainty quantification.

By contrast, our framework provides a middle ground: the LLM does not self-steer probabilistically but delegates uncertainty reasoning to an explicit inference process that is auditable and externally verifiable. This modularity enables controlled reasoning pipelines suitable for high-stakes domains such as medical diagnosis and decision support.

2.5 Positioning and Distinction

Table 1 summarizes key distinctions among prior methods. While existing approaches either rely on implicit stochasticity (e.g., ABC, SMC steering) or natural language verbalization (e.g., vPGM), our framework provides an explicit structured interface between natural language and probabilistic inference. This design avoids fine-tuning, simplifies reproducibility, and allows integration with off-the-shelf probabilistic libraries. By separating semantic parsing from mathematical reasoning, our approach bridges the interpretability of language models with the formalism of Bayesian inference.

out of bound.

Table 1: Comparison of related approaches to probabilistic reasoning in LLMs.

Method	Requires Fine-tuning	Explicit Graphical Model	Formal Inference
ABC-based UQ [8]	No	No	Partial (Approximate)
SMC Steering [11]	Yes	Implicit	Yes (SMC)
Twisted SMC [4]	Yes	Implicit	Yes (Adaptive SMC)
vPGM [2]	No	Verbalized Only	No
Self-Steering LMs [1]	Partial	No	No
Ours (PGM–LLM Framework)	No	Yes (Structured JSON Schema)	Yes (Exact/Approx.)

In summary, our work draws inspiration from probabilistic steering, verbalized reasoning, and uncertainty quantification but contributes a novel synthesis. We treat LLMs as semantic compilers that construct and interpret probabilistic models, while delegating inference to specialized probabilistic engines. This approach unites the interpretability of language-based reasoning with the rigor of formal probabilistic inference.

Too repetitive.

3 Methodology

3.1 Overview

Our proposed framework integrates a ~~Large Language Model (~~LLM~~)~~ with ~~a Probabilistic Graphical Model (~~PGM~~)~~ through a structured *encoding–inference–decoding* pipeline. The design principle is to separate linguistic understanding from probabilistic computation. The LLM serves as a semantic translator that constructs a probabilistic model from user queries, while a dedicated inference engine performs formal Bayesian reasoning. This modularity allows the system to provide transparent, uncertainty-aware explanations without fine-tuning the LLM.

Formally, let q denote a user query expressed in natural language, and $\mathcal{L}$ the LLM. The LLM generates a structured probabilistic representation $G = (V, E, \Theta)$ where V represents random variables (nodes), E the dependency edges, and Θ the set of conditional probability distributions (CPDs). The encoded graph G is passed to an inference module $\mathcal{I}$, which computes posterior probabilities given evidence X_{obs}:

$$P(Y \mid X_{obs}) = \mathcal{I}(G, X_{obs}), \tag{1}$$

where Y denotes the query variable(s). Finally, the result is decoded by the LLM into a natural language explanation r, completing the mapping:

$$r = \mathcal{L}_{decode}(\mathcal{I}(\underbrace{\mathcal{L}_{encode}(q)}_{})). \tag{2}$$

what exactly is this and how does it relate to eq (1)

3.2 Architecture Components

Figure ?? illustrates the overall framework. The system comprises four main components: (1) the *PGM encoder*, (2) the *structural validator*, (3) the *probabilistic inference engine*, and (4) the *PGM decoder*.

22

1. PGM Encoder. The encoder reformulates user queries into structured probabilistic models by prompting the LLM to output JSON objects that define nodes, edges, CPDs, and evidence. For instance:

```
{
  "nodes": ["Pneumonia", "Fever", "Cough"],
  "edges": [["Pneumonia", "Fever"], ["Pneumonia", "Cough"]],
  "cpds": {"Pneumonia": [0.1, 0.9],
          "Fever|Pneumonia": [[0.9, 0.1], [0.2, 0.8]]},
  "evidence": {"Fever": true, "Cough": true},
  "query": "P(Pneumonia|Fever=True,Cough=True)"
}
```

To maintain structural validity, decoding is grammar-constrained using a JSON schema that enforces type consistency, acyclicity, and probability normalization. This approach extends structured prompting methods previously explored by Huang et al. [2] and Zelikman et al. [9].

2. Structural Validator. The validator ensures the syntactic and semantic correctness of the generated PGM. Checks include: (i) acyclicity for Bayesian networks, (ii) valid probability distributions ($\sum p_i = 1$), (iii) matching dimensions for CPDs, and (iv) well-formed evidence. Invalid structures are automatically repaired through a corrective prompt to the LLM, following a constrained self-refinement loop similar to recent tool-augmented LLMs [7].

3. Probabilistic Inference Engine. The inference engine performs the core reasoning step. Given a validated graph G, it computes posterior distributions using either exact or approximate inference:

$$P(Y \mid X_{obs}) = \frac{P(Y, X_{obs})}{\sum_Y P(Y, X_{obs})}. \tag{3}$$

Exact inference uses algorithms such as Variable Elimination or the Junction Tree algorithm [3], while approximate inference uses Sampling, Variational Inference, or Sequential Monte Carlo (SMC) [11, 4]. In this paper's prototype, we use pgmpy and PyMC for inference. This modular approach ensures the framework can support more advanced engines (e.g., Pyro, Gen.jl) without architectural changes.

4. PGM Decoder. The decoder LLM translates the posterior results into an interpretable textual explanation. This step includes: (i) summarizing the computed probability or confidence interval, (ii) describing causal relations (e.g., "Fever increases the likelihood of Pneumonia by 67%"), and (iii) presenting counterfactual reasoning ("If Cough were absent, the probability would drop to 25%"). The decoder enforces factual fidelity by restricting numerical content to values returned from the inference module, avoiding common LLM hallucinations [10].

3.3 Algorithmic Flow

Algorithm 1 summarizes the overall pipeline.

Need to discuss it too, algorithm is not enough.

Algorithm 1 LLM–PGM Integration Framework

```
Require: User query $q$
Ensure: Natural language response $r$
  0: $G \leftarrow \mathcal{L}_{\text{encode}}(q)$ {LLM generates structured PGM}
  0: $G' \leftarrow \text{validate}(G)$ {Acyclic, normalized, well-formed}
  0: $\pi \leftarrow \mathcal{I}(G', X_{\text{obs}})$ {Posterior inference}
  0: $r \leftarrow \mathcal{L}_{\text{decode}}(\pi)$ {Interpret and explain results}
     return $r$ =0
```

3.4 Example: Diagnostic Reasoning

To illustrate, consider the medical query: *"A patient presents with cough and fever. What is the likelihood of pneumonia?"* The encoder constructs a Bayesian network with nodes Pneumonia, Cough, and Fever, linked by directed edges Pneumonia → Cough and Pneumonia → Fever. The inference engine calculates:

$$P(\text{Pneumonia} \mid \text{Cough} = 1, \text{Fever} = 1) = 0.67,$$

and the decoder returns: *"Given the symptoms, the probability of pneumonia is approximately 67%, with a 95% credible interval between 60% and 73%."*

3.5 Advantages

The proposed design offers three main advantages:

1. **Transparency:** Every reasoning step is explicitly represented in the PGM structure, enabling human verification.
2. **Calibration:** Probabilistic inference ensures coherent uncertainty quantification.
3. **Modularity:** The separation of encoding, inference, and decoding allows the system to evolve without retraining the LLM.

Compared with prior probabilistic steering approaches [11, 4], our framework achieves reasoning interpretability with minimal engineering complexity, making it suitable for research and educational settings.

4 Use cases (no results yet)

Diagnostic reasoning Queries like *what is the likelihood of pneumonia given cough and fever* map naturally to a small graph. The interface supports counterfactual toggles for symptoms and tests and can surface which edges drive the posterior.

Causal what if Interventions are expressed by altering evidence or CPDs under a copy of G. The engine recomputes posteriors and the decoder summarizes differences.

Decision support sketch Introducing simple utility nodes enables expected utility comparisons. We outline schema extensions for discrete utilities and decision variables while deferring empirical study.

5 Evaluation plan

Datasets Small reference networks such as Asia and Alarm plus synthetic medical graphs packaged with `pgmpy`. Natural language templates map to these graphs to create paired inputs.

Baselines Chain of Thought prompting, verbalized probabilistic reasoning [2], and probabilistic steering via SMC when applicable [11, 4].

Metrics Accuracy against ground truth posteriors from the reference graphs; calibration via expected calibration error [5]; consistency under prompt paraphrases; interpretability via rubric based human ratings that check reference to edges, evidence, and uncertainty labeling.

Ablations Remove CPD normalization, disable structural validation, vary graph size, and toggle grammar constrained decoding to quantify schema enforcement effects.

Reporting We will release prompts, schemas, and scripts. No numeric results are reported in this draft.

6 Discussion and limitations

Benefits The separation between semantic compilation and numeric inference increases transparency and enables auditing. The minimal contract makes it straightforward to swap backends.

Limitations Exact inference scales poorly with graph size. Approximate inference introduces variance and requires resource management. Prompt sensitivity can cause occasional schema violations that require repair. The present schema targets discrete variables.

Ethical notes Outputs should be treated as decision support. Explanations must label uncertainty clearly to avoid overconfidence in high stakes settings.

7 Future work

Extensions include dynamic and hybrid models, active learning of CPDs, feedback loops that refine G using inference discrepancies, and multi agent settings where several LLMs propose structures reconciled by model selection.

8 Conclusion

We specify a practical bridge between LLMs and PGMs using a minimal schema and an encode validate infer decode pipeline. The design emphasizes validity, transparency, and portability. This draft provides the interfaces and evaluation plan needed for reproducible exploration without reporting empirical outcomes.

References

[1] Benjamin Grand, Tian Zhang, and Jacob Andreas. Self-steering language models. *arXiv preprint arXiv:2501.01234*, 2025.

[2] Wei Huang, Junyi Li, and See-Kiong Ng. Verbalized probabilistic graphical modeling with large language models. *arXiv preprint arXiv:2404.12345*, 2024.

[3] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[4] Qingyu Luo and Andriy Mnih. Probabilistic inference in language models via twisted sequential monte carlo. *arXiv preprint arXiv:2502.04521*, 2025.

[5] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI Conference on Artificial Intelligence*, 2015.

[6] Reza Nafar, Sarthak Sharma, and Yichen Liu. Reasoning over uncertain text by generative large language models. *arXiv preprint arXiv:2503.03210*, 2025.

[7] Timo Schick, Jane Dwivedi-Yu, and et al. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

[8] Sarthak Sharma, Raymond Tang, and Vaishak Belle. Uncertainty quantification of large language models using approximate bayesian computation. *arXiv preprint arXiv:2502.01765*, 2025.

[9] Eric Zelikman, Yuhuai Wu, Noah Goodman, and Maxwell Nye. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 2022.

[10] Rui Zhang, Bill Lin, and Xiaojie Xu. Trustllm: Trustworthiness in large language models. *arXiv preprint arXiv:2402.04513*, 2024.

[11] Liang Zhou, Jiarui Zhang, and Yilin Wang. Sequential monte carlo steering of large language models using probabilistic programs. *arXiv preprint arXiv:2502.06789*, 2025.

Need to write the story better, introduction should flow & have a good motivation. Justify all claims.