# BayesL: Towards a Logical Framework for Bayesian Networks

**Stefano M. Nicoletti**
University of Twente, Enschede, the Netherlands

**Mariëlle Stoelinga**
University of Twente, Enschede, the Netherlands
Radboud University, Nijmegen, the Netherlands

## Abstract

We introduce BayesL, a novel logical framework for specifying, querying, and verifying the behaviour of Bayesian networks (BNs). BayesL (pronounced "Basil") is a structured language that allows for the creation of queries over BNs. It facilitates versatile reasoning concerning causal and evidence-based relationships, and permits comprehensive what-if scenario evaluations without the need for manual modifications to the model.

**2012 ACM Subject Classification** Theory of computation → Logic and verification; Theory of computation → Verification by model checking

**Keywords and phrases** Bayesian networks, Bayesian inference, Logic, Model checking

**Digital Object Identifier** 10.4230/LIPIcs.BMQL.2025.23

**Funding** This work was partially funded by the NWO grant NWA.1160.18.238 (PrimaVera), the EU's Horizon 2020 Marie Curie grant No 101008233, ERC Consolidator and ERC Proof of Concept Grants Nr. 864075 (CAESAR) and Nr. 101187945 (RUBICON).

## 1 Introduction

Bayesian networks (BNs) are a cornerstone model in AI, enabling structured probabilistic reasoning under uncertainty. They model conditional dependencies among random variables using a directed acyclic graph (DAG), with each node representing a random variable and each edge encoding a probabilistic dependency. This structure allows for the compact representation of joint probability distributions [23] enabling efficient inference and enhanced interpretability. BNs have become integral to numerous domains [12].

**Problem statement.** Despite their ubiquitous success, a central challenge remains how to *understand* and *validate* a given BN. Especially when inferred from data, practitioners must ask whether a BN exhibits the properties one expects and how to explain and trace its behaviour. To this end, users require tools that go beyond computing probabilities: they need structured ways to analyse how evidence propagates through the network, to verify whether key dependencies and independencies hold, and to assess the model's response to hypothetical interventions. Constructing *what-if* scenarios is an essential technique in this regard, as it allows users to evaluate the behaviour of the BN under controlled modifications and to check whether the model conforms to the expected behaviour, without the need to manually alter or reparameterize the network. To support these needs, our logic offers a range of reasoning capabilities, including marginal and conditional inference, identification of most likely outcomes, and flexible querying through non-standard comparisons. This enables practitioners to systematically probe the behaviour of the BN across multiple dimensions. In spite of their importance and the extensive body of work on reasoning about BNs (see Sec. 4), BayesL is unique in providing: 1. a flexible logical language for property specification and querying, able to ease the formulation of insightful what-if scenarios without manually

operating on the underlying BN model; 2. a framework grounding automatic analysis via model checking procedures that provide formal guarantees on the behaviour of BNs – ever so crucial in the era of AI; and 3. a reasoning system that integrates Bayesian inference and learning to construct a BN satisfying a given formal specification.

**Our contribution.** To address these needs, we propose a *Bayesian network Logic* (BayesL) – pronounced "Basil" – a query language that allows one to specify logical properties over BNs, through layered reasoning capabilities: graph-structure reasoning with respect to probabilistic influence and conditional independence; flexible probabilistic inference supporting causal and evidential reasoning; and the specification and verification of formal properties of BN behaviour. BayesL further supports the formulation of rich what-if scenarios without requiring manual updates to the underlying BN model, and is designed to operate even in the presence of partially specified BNs, whether incomplete in topology or parameters: we envision future extensions that will leverage advances in probabilistic model checking and program synthesis [18, 19, 20, 1] to support learning and constrained synthesis of BNs that satisfy formal BayesL constraints.

In this work-in-progress contribution, we focus on defining the expressive capabilities of BayesL and its underlying syntax and semantics, laying the foundation for future work on model checking algorithms, model synthesis, and tool development.

## 2 Background on Bayesian Networks

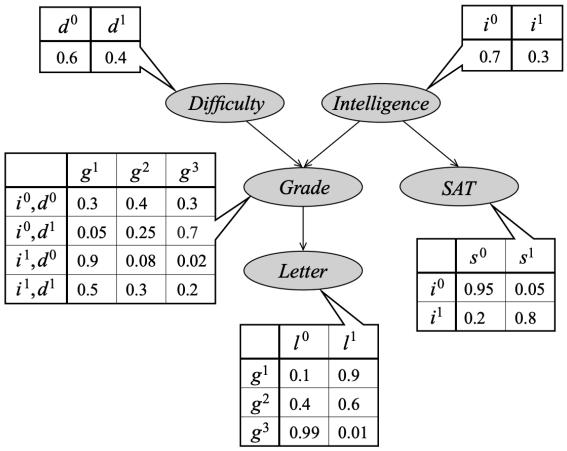


**Figure 1** A simple BN representing dependencies between the quality of a recommendation letter, the SAT score, the grade obtained in a course, the difficulty of that course and the intelligence of a student, from [12]. The image includes conditional probability tables for every node in the BN.

A BN is a tuple $B = (G, \Theta)$ where $G = (V, \to)$ is a directed acyclic graph with a finite set of vertices $V$. Each $v \in V$ represents a random variable $X_v : \Omega \to D$ taking values in a finite domain $D$. We often identify $v$ and $X_v$, and write $X$ for $X_v$. Each edge $v \to w$ represents the dependency of $X_v$ on $X_w$. We let $\mathit{parents}(v) = \{w \in V \mid w \to v\}$ be the set of incoming vertices to $v$. Each vertex $v$ is equipped with a *conditional probability table* (CPT) $\Theta_v$ that expresses the probability distribution of $X_v$ in terms of its parents: for a vertex $v$ with parents $w_1, \dots w_k$, the function $\Theta_v : D^{k+1} \to [0, 1]$ represents the conditional probability distribution $\Theta_v(d_1, \dots d_k)(d) = Pr(X_v = d \mid X_{w_1} = d_1, \dots, X_{w_k} = d_k)$. We then let $\Theta : \bigsqcup_{v \in V} D^{|\text{parents}(v)|+1} \to [0, 1]$ denote the BN CPD, where $v$ identifies the node, and the restriction $\Theta_v := \Theta|_v$ recovers the original $\Theta_v : D^{|\text{parents}(v)|+1} \to [0, 1]$.

▶ **Example 1.** Fig. 1 represents a well-known BN for the student example from [12]. This BN encodes dependencies between five random variables: the student's intelligence (`Int`), the course difficulty (`Dif`), the grade (`Gra`), the student's SAT score (`SAT`), and the quality of the recommendation letter they obtain (`Let`). All of the variables except `Gra` are binary-valued, and `Gra` is ternary-valued (i.e., grade is either *high*, *medium* or *low*). Fig. 1 also represents the CPTs of each vertex representing said variables: the CPT of vertex $v$ defines a probability distribution which determines the evaluation of $v$, given some evaluation of *parents*$(v)$ [23]. E.g., according to the CPT of *Grade*, the probability of getting a *medium* grade ($\texttt{Gra} = \texttt{g}^2$) given a *difficult* exam ($\texttt{Dif} = \texttt{d}^1$) and an *intelligent* student ($\texttt{Int} = \texttt{i}^1$) is 0.3. The semantics of a BN $B = (G, \Theta)$ is thus the joint probability function that it defines [23].

A central task when working with Bayesian networks (BNs) is *inference*, that is, computing the probability of one or more random variables given some known evidence about other variables in the network. Several core types of inference are typically considered (and supported by BayesL):

- **Marginal inference.** Compute the probability of a variable by summing out all other variables from the joint distribution. This provides an overall belief about the variable without conditioning on evidence. E.g., $\mathrm{P}(\texttt{Let} = l^1)$.
- **Conditional inference.** Compute the probability of a variable given known evidence about other variables in the network. This is the most common form of query used in diagnostic, predictive, or explanatory reasoning [12]. E.g., $\mathrm{P}(\texttt{Let}{=}l^1 \mid \texttt{Gra}{=}g^1, \texttt{Dif}{=}d^1)$.
- **Marginal Maximum A Posteriori.** A cornerstone operator in BN inference, this query finds the most probable assignment of a subset of variables given evidence. This task is often used in decision-making or classification settings where one seeks the most likely explanation for observations with respect to specific variables.
- **MPE (Most Probable Explanation).** This query finds the most probable complete assignment to *all* variables in the network, given evidence. This is used when a fully specified scenario or explanation is needed based on the available evidence.

# 3 BayesL: a Bayesian network Logic

## 3.1 BayesL Queries

Before giving the formal syntax and semantics, we illustrate BayesL via some example queries for the BN in Fig. 1. These examples illustrate how BayesL supports querying a BN at multiple levels (incl. non-standard ones):

1. **Evidential reasoning.** Given that the *Letter* is *weak*, how likely is it that the course was *difficult*?
$$\mathrm{P}(\texttt{Dif} = \texttt{d}^1 \mid \texttt{Let} = \texttt{l}^0)$$
2. **Causal reasoning with disjunction.** How likely is a *strong* recommendation *Letter* because of a *high SAT* score or a *high Grade*?
$$\mathrm{P}(\texttt{Let} = \texttt{l}^1 \mid \texttt{SAT} = \texttt{s}^1 \vee \texttt{Gra} = \texttt{g}^1)$$
3. **Non-standard comparison.** What is the probability that the student obtains a *Grade* lower than *high* or that the course was *easy*?
$$\mathrm{P}(\texttt{Gra} < \texttt{g}^1 \vee \texttt{Dif} = \texttt{d}^0)$$

4. **Boolean combination & threshold checking.** Is it true that the probability of a *strong Letter* given a *high Grade* is at least 0.8 and that *Intelligence* and *Letter* are independent given *Grade*?

$$\mathrm{P}(\texttt{Let} = \texttt{l}^1 \mid \texttt{Gra} = \texttt{g}^1) \geq 0.8 \wedge \mathrm{IDP}(\texttt{Int}, \texttt{Let} \mid \texttt{Gra})$$

5. **What-if/CPT update.** After updating the probability of a *high Grade* given *high Intelligence* and a *difficult* course to 0.9, does the probability of a *strong Letter* improve beyond 0.7?

$$\mathrm{P}(\texttt{Let} = \texttt{l}^1 \mid \texttt{Gra} = \texttt{g}^1) \geq 0.7\ [\texttt{Gra} = \texttt{g}^1 \mid \texttt{Int} = \texttt{i}^1, \texttt{Dif} = \texttt{d}^1 \mapsto 0.9]$$

6. **Marginal Maximum A Posteriori query.** What is the most probable assignment to *Intelligence* and *SAT* given a *strong Letter*?

$$\mathrm{MAP}(\texttt{Int}, \texttt{SAT} \mid \texttt{Let} = \texttt{l}^1)$$

7. **Most Probable Explanation query.** What is the most probable complete explanation for the observed *strong Letter*?

$$\mathrm{MPE}(\texttt{Let} = \texttt{l}^1)$$

## 3.2 BayesL Syntax

**Atoms.** At the base of BayesL syntax are atomic "*events*", i.e., formulae of the form $X \bowtie x$, where $X \in V$ is a single variable, $x \in D$ is an assignment on that variable, and $\bowtie \in \{<, \leq, =, \geq, >\}$ denotes a comparison operator (that must be allowed) on the value domain $D$. Atomic formulae can be combined via Boolean connectives to form non-standard queries, enabling logical expressions such as $(X_1 \leq x_1) \wedge (X_2 = x_2)$. Where convenient, we also use the notation $\overline{X} = (X_1, \ldots, X_k)$ and $\overline{x} = (x_1, \ldots, x_k)$ to denote ordered vectors of variables and corresponding values.

**Layer 1: Probabilistic inference.** Layer 1 supports standard probabilistic reasoning tasks. The construct $\mathrm{P}(\alpha \mid \alpha)$ denotes the standard conditional probability between two (Boolean) formulae in Atoms. Marginal queries are treated as a special case with omitted condition, i.e., $\mathrm{P}(\alpha)$ denotes the marginal probability of $\alpha$. This layer also supports *marginal maximum a posteriori* (MAP) – finds the most probable assignment of a subset of variables given evidence – and *most probable explanation* (MPE) inference – finds the most probable complete assignment to all variables in the network, given evidence. Specifically, $\mathrm{MAP}(\overline{X} \mid \alpha)$ returns the set of assignments to $\overline{X}$ that maximize the conditional probability $\Pr(\overline{X} = \overline{x} \mid \alpha)$, while $\mathrm{MPE}(\alpha)$ returns the set of full assignments to non-evidence variables in $\alpha$ that maximize that probability. To support hypothetical reasoning, Layer 1 includes a local CPT update operator $\phi\,[X = x \mid \overline{E} = \overline{e} \mapsto q]$, which allows evaluating how probabilities change under modifications to specific entries in the BN's conditional probability tables. This operator updates the entry for $X = x$ given $\overline{E} = \overline{e}$ to $q$, and renormalizes the remaining row values accordingly. When concatenating multiple of these operators, precedence is crucial to guarantee coherent updates on CPTs: e.g., if the same field of a CPT is updated multiple times, then the innermost update will overwrite the others. Here, we fix $X$ to be a single variable and $\overline{E}$ a vector of its parents. Each Layer 1 formula evaluates to either a probability value in $[0, 1]$ – for $\mathrm{P}(\cdot)$ and update formulae – or a set of assignments – for MAP and MPE queries. Together, these constructs provide a rich and compositional language for expressing both standard and non-standard probabilistic queries.

**Layer 2: Probabilistic constraints.** Layer 2 enables reasoning about probabilistic assertions and logical combinations thereof. The main construct $\phi \bowtie p$ compares the result

of a Layer 1 query against a threshold $p \in [0,1]$, using any standard comparison operator $\bowtie$. This layer also includes Boolean connectives for forming composite queries, as well as the same CPT update operator used in Layer 1, now lifted to allow evaluating the impact of updates on formulae of this layer: $\psi\,[X = x \mid \overline{E} = \overline{e} \mapsto q]$ expresses the update of the $\psi$ formula via $[X = x \mid \overline{E} = \overline{e} \mapsto q]$ before its evaluation.

**Layer 3: Structural reasoning.** Layer 3 captures structural properties of the BN, independent of parameter values (i.e., only considering the graph structure $G$). Given a set of variables $\overline{E}$, we say that an *active trail* between two nodes $X$ and $Y$ exists if there is a path $\pi$ between $X$ and $Y$ in $G$ such that every triple along $\pi$ satisfies the following [12]:

- For every *chain* or *fork* structure $A \to B \to C$ or $A \leftarrow B \to C$, the node $B$ is not in $\overline{E}$,
- For every *v-structure* $A \to B \leftarrow C$, either $B \in \overline{E}$ or some descendant of $B$ is in $\overline{E}$.

If no such trail exists, then $X$ and $Y$ are *d-separated* given $\overline{E}$, i.e., structurally conditionally independent in all compatible distributions. The formula $\mathrm{INFL}(X, Y \mid \overline{E})$ holds if there exists an active trail from $X$ to $Y$ given $\overline{E}$, indicating possible influence propagation. Conversely, $\mathrm{IDP}(X, Y \mid \overline{E})$ holds if $X$ and $Y$ are d-separated given $\overline{E}$. As in the other layers, structural formulae support Boolean combinations through negation and conjunction.

$$
\begin{aligned}
&\text{Atoms:} && \alpha ::= X \bowtie x \mid \neg\alpha \mid \alpha \wedge \alpha && \text{with } \bowtie \in \{<, \leq, =, \geq, >\}\\
&\text{Layer 1:} && \phi ::= \mathrm{P}(\alpha \mid \alpha) \quad \phi' ::= \phi\,[X = x \mid \overline{E} = \overline{e} \mapsto q] \mid \mathrm{MAP}(\overline{X} \mid \alpha) \mid \mathrm{MPE}(\alpha)\\
&\text{Layer 2:} && \psi ::= \phi \bowtie p \mid \psi\,[X = x \mid \overline{E} = \overline{e} \mapsto q] \mid \neg\psi \mid \psi \wedge \psi\\
&\text{Layer 3:} && \gamma ::= \mathrm{INFL}(X, Y \mid \overline{E}) \mid \mathrm{IDP}(X, Y \mid \overline{E}) \mid \neg\gamma \mid \gamma \wedge \gamma \mid \psi \wedge \gamma
\end{aligned}
$$

## 3.3 BayesL Semantics

**Atoms semantics.** Formulae in the Atoms layer are evaluated over full assignments $a \in D^V$, which we interpret as functions $a : V \to D$ assigning a value to each variable, and a BN $B$. The semantics $[\![\alpha]\!]_B(a) \in \{0, 1\}$ is defined recursively as follows:

$$
\begin{aligned}
[\![X \bowtie x]\!]_B(a) &= 1 \quad \text{iff } a(X) \bowtie x\\
[\![\neg\alpha]\!]_B(a) &= 1 \quad \text{iff } [\![\alpha]\!]_B(a) = 0\\
[\![\alpha_1 \wedge \alpha_2]\!]_B(a) &= 1 \quad \text{iff } [\![\alpha_1]\!]_B(a) = 1 \text{ and } [\![\alpha_2]\!]_B(a) = 1
\end{aligned}
$$

We treat atoms as Boolean-valued predicates over assignments. This semantics supports arbitrary comparison operators $\bowtie \in \{<, \leq, =, \geq, >\}$, and thus accommodates non-standard comparisons such as inequalities.

**Layer 1 semantics.** Formulae in Layer 1 are evaluated over a BN $B = (G, \Theta)$ and return either a probability in $[0, 1]$ or a set of assignments (for MAP and MPE queries). Let $\Pr_B(a)$ denote the joint probability of a full assignment $a \in D^V$ under $B$, computed via the standard BN factorization [12]:

$$
\Pr_B(a) = \prod_{v \in V} \Theta_v\Big(a(\mathit{parents}(v))\Big)\Big(a(v)\Big)
$$

The semantics of Layer 1 is thus defined as follows:

$$
\begin{aligned}
[\![\mathrm{P}(\alpha)]\!]_B &= \sum_{a \in D^V} [\![\alpha]\!]_B(a) \cdot \Pr_B(a)\\
[\![\mathrm{P}(\alpha_1 \mid \alpha_2)]\!]_B &= \frac{[\![\mathrm{P}(\alpha_1 \wedge \alpha_2)]\!]_B}{[\![\mathrm{P}(\alpha_2)]\!]_B} \quad \text{with } [\![\mathrm{P}(\alpha_2)]\!]_B > 0\\
[\![\mathrm{MAP}(\overline{X} \mid \alpha)]\!]_B &= \operatorname*{argmax}_{\overline{x} \in D^{|\overline{X}|}} \Pr_B(\overline{X} = \overline{x} \mid \alpha)
\end{aligned}
$$

$$[\![\mathrm{MPE}(\alpha)]\!]_B = \operatorname*{argmax}_{\overline{v} \in D^{|\overline{V \setminus E}|}} \Pr_B(\overline{V \setminus E} = \overline{v} \mid \alpha)$$

where $\overline{V \setminus E}$ refers to an ordered vector of the variables in $V \setminus E$, with $\overline{v} \in D^{|\overline{V \setminus E}|}$ denoting corresponding assignments. Here, $\overline{E}$ denotes the vector of variables constrained in the evidence formula $\alpha$.

**CPT updates.** The update expression $\phi\,[X = x \mid \overline{E} = \overline{e} \mapsto q]$ evaluates the Layer 1 formula $\phi$ over the updated BN $B[q/(X = x \mid \overline{E} = \overline{e})] = (G, \Theta')$, where:

$$\Theta'_v = \Theta_v \quad \text{for all } v \neq X$$

$$\Theta'_X(\overline{e})(x') = \begin{cases} q & \text{if } x' = x \\ \Theta_X(\overline{e})(x') \cdot \dfrac{1-q}{1-\Theta_X(\overline{e})(x)} & \text{otherwise} \end{cases}$$

This guarantees that the modified row $\Theta'_X(\overline{e})$ remains a valid distribution over $D$.

**Layer 2 semantics.** Formulae in Layer 2 are evaluated over a BN $B = (G, \Theta)$ and return Boolean values. They extend Layer 1 with threshold comparisons, logical combinations, and perform CPT updates. With Booleans resolved as usual, we let semantics of $\psi$-formulae be:

$$[\![\phi \bowtie p]\!]_B = 1 \quad \text{iff } [\![\phi]\!]_B \bowtie p; \qquad [\![\psi[X = x \mid \overline{E} = \overline{e} \mapsto q]]\!]_B = [\![\psi]\!]_{B[q/(X=x|\overline{E}=\overline{e})]}$$

**Layer 3 semantics.** Formulae in Layer 3 express properties of the BN's structure and are evaluated solely on the graph $G$: i.e., they ignore $\Theta$. We define:

$$[\![\mathrm{INFL}(X, Y \mid \overline{E})]\!]_B = 1 \quad \text{iff there exists an } \textit{active trail} \text{ from } X \text{ to } Y \text{ given } \overline{E} \text{ in } G$$
$$[\![\mathrm{IDP}(X, Y \mid \overline{E})]\!]_B = 1 \quad \text{iff } X \text{ and } Y \text{ are } \textit{d-separated} \text{ given } \overline{E} \text{ in } G$$

## 4 Related work

Several frameworks have been developed to facilitate reasoning on (models similar to) BNs:
1. Frameworks that allow reasoning on BNs: [9, 10, 2, 5, 7, 3, 15, 14, 17, 25, 4, 24];
2. Model checking techniques for reasoning on BNs or similar models: [23, 18, 19, 20].

### 4.1 Tools and Frameworks for Bayesian Reasoning

**Bayesian reasoning and channels:** [10] offers a language for describing Bayesian phenomena in terms of programming concepts – like channel and predicate transformation – modelling inference as a calculus of string diagrams. Based on these semantics, [9] proposes a new algorithm for exact Bayesian inference. However, these works do not offer infrastructure taylored towards automatic verification of BNs' formal properties via model checking.

**Bayesian Logic Programs (BLPs):** BLPs combine BNs with definite clause logic, establishing a one-to-one mapping between ground atoms and random variables [11]: this allows for the representation of objects and relations, overcoming some limitations of propositional logic. However, BLPs primarily focus on the integration of logic programming with probabilistic reasoning: BayesL aims to provide verification capabilities and expressivity while remaining at the level of the BN for property specification, giving practitioners a method to query the BN model directly and avoiding lower-level querying on an intermediate translation model.

**Probabilistic Soft Logic (PSL):** PSL combines first-order logic with probabilistic graphical models [2] and supports efficient inference through convex optimization. However, PSL is focussed on modelling rich, structured data, esp. via the definition of hinge-loss Markov random fields and does not provide infrastructure for formal verification of BN properties.

**CLP(BN):** Constraint Logic Programming for Bayesian Networks (CLP(BN)) expresses BNs through the constraint logic programming framework [5]. It addresses capturing of probabilistic data using logic-based representations. CLP(BN) thus constitutes an extension of logic programming focused on defining joint probability distributions, and does not address verification of formal properties of BNs.
**GUIs/APIs:** Tools like GeNIe/SMILE [7] and Bayes Server [3] provide graphical interfaces and APIs for BN modeling and inference. These tools support various functionalities, including learning, inference, and visualization. However, they lack capabilities for formal specification and verification of BN behaviour.
**BNMonitor:** The `bnmonitor` [15] R package allows for sensitivity and robustness analysis in BNs, and assessment of the impact of changes in the network structure or parameters. However, it does not provide a logical language for specifying and verifying formal properties.
**Multi-Entity Bayesian Networks (MEBN):** MEBN combines BNs with first-order logic to represent complex domains involving multiple entities and relationships [14]. While MEBN enhances the expressiveness of BNs, it adopts a proof-theoretial approach and focuses on knowledge representation, rather than on formal specification and automatic verification oriented towards model checking.
**OpenBUGS and Bambi:** Tools like OpenBUGS [17] and Bambi [25] facilitate Bayesian analysis using (probabilistic) programming. They allow for model specification and inference but do not offer mechanisms for formal verification of BN properties.
**ShinyBN:** ShinyBN is an R/Shiny application that provides an interactive interface for BN modeling and inference [4]. While it enhances usability, it does not support formal specification and verification.
**BNLearn:** `bnlearn` [24] is an R package for learning the graphical structure of BNs, estimating their parameters and performing probabilistic and causal inference. In spite of the extensive feature set, it does not provide a logical language for formal verification.

## 4.2 Probabilistic Model Checking Approaches

**A model checking approach:** Salmani and Katoen [23] present a method that translates BNs into tree-like Markov chains, enabling inference through reachability probabilities. This methodology aims to levereage probabilistic model checking techniques and established tooling, e.g., Storm [6] and Prism [13] model checkers. While this approach allows for formal verification, it requires the transformation of BNs into different structures and property specification is done at the level of the underlying Markov structure (with e.g., PLTL [21] and PCTL [8]). BayesL aims to provide verification capabilities and expressivity while remaining at the level of the BN for property specification: a methaporical higher-level language for BN querying, w.r.t. languages that query the underlying lower-level model translations.
**BFL, PFL and ATM.** Recent work develops logics for reasoning on *fault trees* [18, 19] and *attack trees* [20]. These contributions do not allow for reasoning on BNs, however are aligned in intent to us: i.e., they enable flexible reasoning and checking via a logic expressing queries directly on the model of interest, and not on its underlying (Markovian) representation.

# 5 Conclusion

This work-in-progress contribution introduces *BayesL*, a logic designed to enhance the interpretability and formal reasoning capabilities on BNs. BayesL supports expressive querying over BNs, including *conditional independence* and *probabilistic influence* analysis, as well as *causal* and *evidential* reasoning. A key feature is the ability to formulate *what-if* scenarios via local updates to CPT entries, without manual modification of the underlying model. The logic further supports *non-standard queries* involving threshold comparisons

and flexible Boolean combinations of atomic properties, allowing users to express rich and compositional probabilistic queries. In doing so, BayesL provides a principled framework for the *formal specification and verification* of BN behaviour. Future work will focus on developing scalable *model checking algorithms* for BayesL – reflecting semantics but encoding them in a computationally efficient approach [16, 22], and on integrating *constraint-guided model learning* to enable the synthesis of BNs that provably satisfy given logical requirements, even when starting from partially specified models [1].

## References


1 Roman Andriushchenko, Milan Češka, Sebastian Junges, Joost-Pieter Katoen, and Šimon Stupinskỳ. PAYNT: a tool for inductive synthesis of probabilistic programs. In *International Conference on Computer Aided Verification*, pages 856–869. Springer, 2021.

2 Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-Loss Markov Random Fields and Probabilistic Soft Logic. *Journal of Machine Learning Research*, 18(109):1–67, 2017. URL: `https://jmlr.org/papers/volume18/15-631/15-631.pdf`.

3 Bayes Server. Bayes Server: Overview, 2025. Accessed: 2025-06-06. URL: `https://bayesserver.com/docs/`.

4 Jiajin Chen, Ruyang Zhang, Xuesi Dong, Lijuan Lin, Ying Zhu, Jieyu He, David C Christiani, Yongyue Wei, and Feng Chen. shinyBN: an online application for interactive Bayesian network inference and visualization. *BMC bioinformatics*, 20:1–5, 2019.

5 Vítor Santos Costa, David Page, Maleeha Qazi, and James Cussens. CLP(BN): Constraint Logic Programming for Probabilistic Knowledge. *arXiv preprint arXiv:1212.2519*, 2012. URL: `https://arxiv.org/abs/1212.2519`.

6 Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In *International Conference on Computer Aided Verification*, pages 592–600. Springer, 2017.

7 Marek J Druzdzel. Smile: Structural modeling, inference, and learning engine and genie: a development environment for graphical decision-theoretic models. In *Aaai/Iaai*, pages 902–903, 1999.

8 Hans Hansson and Bengt Jonsson. A Logic for Reasoning about Time and Reliability. *Formal Aspects Comput.*, 6(5):512–535, 1994. `doi:10.1007/BF01211866`.

9 Bart Jacobs. A channel-based exact inference algorithm for bayesian networks. *arXiv preprint arXiv:1804.08032*, 2018.

10 BPF Jacobs, Fabio Zanasi, et al. *The logical essentials of Bayesian reasoning*. Cambridge University Press, 2021.

11 Kristian Kersting and Luc De Raedt. Bayesian Logic Programs. *arXiv preprint cs/0111058*, 2001. URL: `https://arxiv.org/abs/cs/0111058`.

12 Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

13 Marta Kwiatkowska, Gethin Norman, and David Parker. Prism: Probabilistic symbolic model checker. In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 200–204. Springer, 2002.

14 Kathryn B Laskey. MEBN: A Language for First-Order Bayesian Knowledge Bases. *Artificial Intelligence*, 172(2-3):140–178, 2008. URL: `https://www.sciencedirect.com/science/article/pii/S0004370207001312`.

15 Manuele Leonelli, Ramsiya Ramanathan, and Rachel L Wilkerson. Sensitivity and robustness analysis in Bayesian networks with the bnmonitor R package. *Knowledge-Based Systems*, 278:110882, 2023.

16 Carlo Lucibello, Fabrizio Pittorino, Gabriele Perugini, and Riccardo Zecchina. Deep learning via message passing algorithms based on belief propagation. *Machine Learning: Science and Technology*, 3(3):035005, 2022.

**17** David Lunn, David Spiegelhalter, Andrew Thomas, and Nicky Best. The BUGS Project: Evolution, Critique and Future Directions. *Statistics in Medicine*, 28(25):3049–3067, 2009.

**18** Stefano M Nicoletti, E Moritz Hahn, and Mariëlle Stoelinga. BFL: a Logic to Reason About Fault Trees. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 441–452. IEEE, 2022.

**19** Stefano M Nicoletti, Milan Lopuhaä-Zwakenberg, E Moritz Hahn, and Mariëlle Stoelinga. PFL: A Probabilistic Logic for Fault Trees. In *International Symposium on Formal Methods*, pages 199–221. Springer, 2023.

**20** Stefano M Nicoletti, Milan Lopuhaä-Zwakenberg, Ernst Moritz Hahn, and Mariëlle Stoelinga. ATM: A Logic for Quantitative Security Properties on Attack Trees. In *International Conference on Software Engineering and Formal Methods*, pages 205–225. Springer, 2023.

**21** Zoran Ognjanovic. Discrete Linear-time Probabilistic Logics: Completeness, Decidability and Complexity. *J. Log. Comput.*, 16(2):257–285, 2006. doi:10.1093/logcom/exi077.

**22** Kristian G Olesen and Anders L Madsen. Maximal prime subgraph decomposition of bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 32(1):21–31, 2002.

**23** Bahare Salmani and Joost-Pieter Katoen. Bayesian inference by symbolic model checking. In *Quantitative Evaluation of Systems: 17th International Conference, QEST 2020, Vienna, Austria, August 31–September 3, 2020, Proceedings 17*, pages 115–133. Springer, 2020.

**24** Marco Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of statistical software*, 35:1–22, 2010.

**25** Tal Yarkoni and Jake Westfall. Bambi: A simple interface for fitting Bayesian mixed effects models. 2016.