Probabilistic Programming: Semantics and Synthesis

Confirmation of Status

Younesse Kaddar

Supervisor: Sam Staton

Christ Church College

Abstract

My research is about probabilistic programming semantics and synthesis. More specifically, it is about the semantics of probabilistic programming for Bayesian nonparametrics, and the integration of probabilistic programming and large language models (LLMs) trained as amortized samplers (with Generative Flow Networks) for safer and more interpretable AI systems. The extended report will be organized in two parts: theoretical foundations on the one hand, and practical applications with probabilistic program synthesis on the other. Part I is theoretical and develops a categorical semantics for LazyPPL, a Haskell-based library leveraging lazy evaluation for infinite-dimensional Bayesian models (e.g., Poisson processes, Gaussian processes, Dirichlet processes, etc.). Using quasi-Borel spaces and presheaf categories, we formalize properties like compositionality and stochastic memoization, providing a categorical framework for compositional probabilistic reasoning. Part II, more applied, introduces SynthStats, a new probabilistic program synthesis framework that leverages GFlowNet-finetuned LLMs to do amortized sampling from distributions over statistical models (expressed as probabilistic programs), given a natural language description. Initial experiments are promising. This line of research posits that distribution-based synthesis of statistical models overcomes key limitations of "black-box" deep learning solutions, contributing to more trustworthy AI through theory and applications.

Confirmation of Status: General Overview

Overview

Large language models (LLMs) show impressive capabilities across a broad range of tasks; however, getting guarantees about their behaviour, particularly in safety-critical settings, remains an open problem. Three key obstacles are commonly identified:

(i) **Opacity/"black-box" aspect**: The internal reasoning processes of LLMs are difficult to inspect directly, hindering interpretability.

(ii) **Uncertainty miscalibration**: Model outputs lack reliable confidence scores, making it hard to gauge whether the model's predictions and assertions are trustworthy.

(iii) **Informal semantics**: LLM outputs exist in unstructured token spaces (text, code, images, etc.), lacking the formal semantics often required for rigorous checking or verification.

Probabilistic programming languages (PPLs) [Mee+18], by contrast, offer features that address some of these limitations. PPLs provide an explicit formal language for generative models, principled methods for uncertainty quantification, and mathematically precise semantics [Sta+17; Ste21; VKS19].

In many ways, LLMs and PPLs have complementary strengths and weaknesses. LLMs excel at leveraging (a) *rich, amortised priors* acquired from vast pre-training data (e.g., Llama 3 was trained on ∼15 trillion tokens, equivalent to roughly $4 \cdot 10^5$ years of human reading time [Gra+24]), (b) understanding *natural language* instructions, and (c) performing *on-the-fly heuristic reasoning* approximating System 2-style deliberation [Kah12] via chain-of-thought (CoT) [Wei+22], particularly in RL-post-trained variants (e.g., OpenAI o-series [Ope+24], DeepSeek R1 [Dee+25], Gemini "Thinking" [Goo25], Claude 3.7 Sonnet Thinking [Ant25]). However, they suffer from the opacity, uncertainty miscalibration, and lack of formal semantics mentioned earlier. Conversely, PPLs express explicit causal structures, support sound statistical inference, and possess formal, compositional programming language semantics [Sta+17; Ste21; VKS19]. Their main drawback is that crafting high-quality models typically requires domain expertise and significant manual effort. This complementarity motivates a hybrid approach: leveraging LLMs' generative power and world knowledge to propose probabilistic programs, while relying on the structure and semantics of PPLs to ensure the interpretability and "auditability" of the resulting statistical models.

My DPhil research explores such a hybrid approach, developing both the theoretical foundations for expressive PPLs (Part I of the planned thesis) and a framework for LLM-guided synthesis of PPL programs (Part II). The central idea is to use LLMs, guided by deep learning amortized inference techniques like Generative Flow Networks (GFlowNets) [Ben+21], not as unrestricted agents or opaque oracles providing final untrustworthy answers, but

Aspect	LLMs (Large Language Models)	PPLs (Prob. Prog. Languages)
Strengths	• Rich natural language priors • Massive compute amortized • Chain-of-thoughts reasoning	• Formal language and semantics • Sound statistical inference • Compositional, interpretable structure
Weaknesses	• Black-box (lack of interpretability) • Uncertainty miscalibration • No formal semantics	• Requires expert domain knowledge • High effort to write complex models • Expressiveness constrained by the domain-specific language

Table 1: Complementary strengths and weaknesses of LLMs and PPLs.

as generators of structured, interpretable statistical models (expressed as probabilistic programs) whose meaning can be formally analyzed. PPLs are chosen as the target language because they offer a convenient balance: their formal semantics provide structure, while their expressiveness is sufficient for complex statistical modelling (subsuming probabilistic graphical models [KF09], for example).

Part I of the planned thesis establishes semantic foundations centred on categorical probability, LazyPPL, and stochastic memoization. Part II investigates LLM-guided synthesis using GFlowNets, leading to our SynthStats framework.

Theoretical Foundations: Semantics for Expressive PPLs (Planned Thesis Part I)

Part I presents foundational work on the semantics of probabilistic programming, focusing on concepts relevant to Bayesian nonparametric (potentially infinite-dimensional) models. Bayesian nonparametric models, like Gaussian Processes (random functions) or Dirichlet Processes (clustering with an unbounded number of classes), are a powerful approach to statistical learning: contrary to parametric models, they can have an unbounded number of parameters that grows dynamically to fit the data.

(i) **LazyPPL (POPL 2023) [Das+23; Laz25].** In collaboration with Staton, Dash, and Paquet, I contributed to the development of LazyPPL, a Haskell library leveraging lazy evaluation for Bayesian nonparametrics. LazyPPL's design uses two monads: an *affine* probability monad Prob (of normalized distributions, for sampling) and a *non-affine* measure monad Meas (of unnormalized distributions, for making observations). Soundness is established via quasi-Borel spaces (QBS) categorical semantics. This enables compositional expression of infinite structures in a declarative manner. For example, an unbounded Poisson Point Process (PPP) with rate λ can be defined recursively as:

```
poissonPP :: ℝ → ℝ → Prob [ℝ]
poissonPP t₀ λ = do
  step ← exponential λ  -- Sample time to next event
  let t = t₀ + step     -- Calculate event time
  ts ← poissonPP t λ    -- Recursively generate rest (lazily)
  return (t : ts)       -- Prepend to infinite stream
```

Laziness ensures only the points that are actually needed are computed (for example, by the viewport, for plotting, at runtime), avoiding manual truncation required in eager PPLs. This facilitates composition: e.g., building a piecewise linear regression model by combining the PPP (for change points) with an infinite lazy stream (iid) of functions from an arbitrary prior distribution.

(ii) **Presheaf Semantics for Stochastic Memoization (MFPS 2023) [KS23].** Deterministic memoization caches function results for efficiency, but does not alter semantics. *Stochastic* memoization, on the other hand, by ensuring consistent random choices for repeated function calls with the same argument, does change the programming language semantics, because it enables us to express infinite random structures in Bayesian nonparametric models [Woo+09]. However, naively implementing memoization with state breaks the crucial *dataflow property* (allowing reordering of independent computations), which is one of the tenets

of probability. In joint work with Staton, we developed a novel an operational and categorical semantics to stochastic memoization and name generation in the context of a minimal probabilistic programming language, that correctly handles stochastic memoization while preserving dataflow property. The core construct is a probabilistic local state monad T on the category $[\mathbf{BiGrph}_{emb}, \mathbf{Set}]$ of presheaves on finite bipartite graphs g (representing memo-table states), defined as:

$$T(X)(g) \;:=\; \left(P_{\mathrm{f}} \int^{g \hookrightarrow h} \left(X(h) \times [0,1]^{(h-g)_L} \right) \right)^{[0,1]^{g_L}}$$

This definition models computations that: depend on an initial state (exponent $[0,1]^{g_L}$); can generate new names/functions (coend $\int^{g \hookrightarrow h}$) probabilistically ($P_{\mathrm{f}}$); produce results ($X(h)$); and assign state to new functions ($[0,1]^{(h-g)_L}$). The monad structure ensures these effects compose correctly while preserving dataflow. I also implemented a Haskell test-bed verifying this model empirically [KS23].

(iii) **Graphons and PPL Interfaces (POPL 2024) [Ack+24].** In further collaborative work, led primarily by Staton, we established a correspondence between equational theories for random graph-generating PPL interfaces and graphons. My contribution focused primarily on the *equational-theory-to-graphon* direction of the proof, which involved arguments based on exchangeability [Ack+24].

Collectively, these results provide semantic principles for expressive nonparametric PPLs, which will be detailed in the full thesis.

LLM-Guided Probabilistic Program Synthesis: The SynthStats Framework (Planned Thesis Part II)

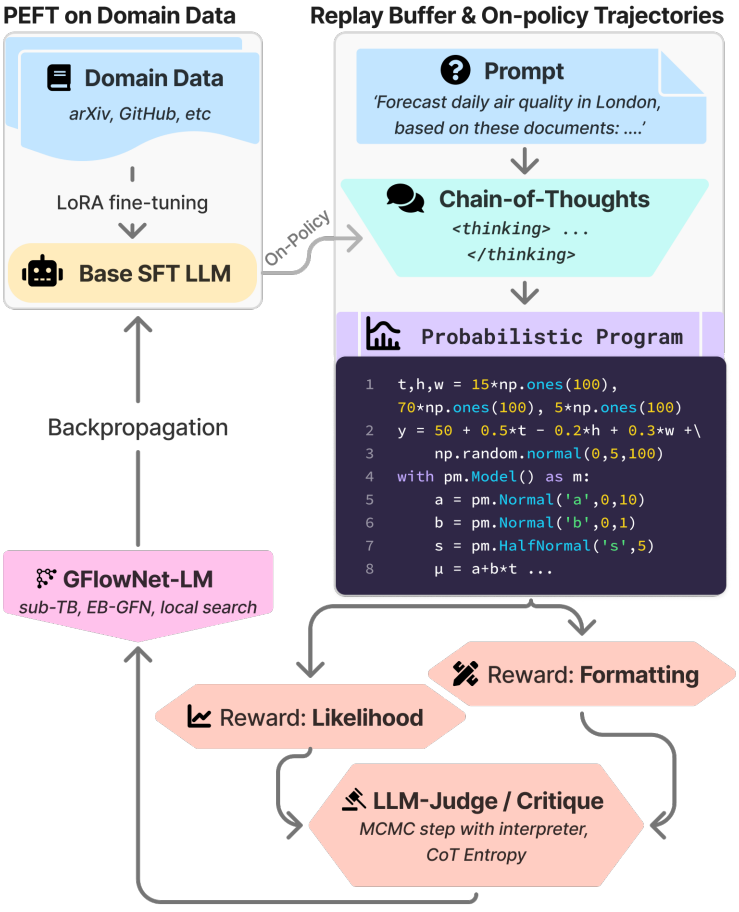

Figure 1: High-level SynthStats pipeline. An LLM finetuned by a GFlowNet (amortized sampler) samples from a distribution over probabilistic programs based on rewards.

Part II addresses the challenge of automatically generating PPL programs from high-level natural language specifications. The proposed framework, **SynthStats**, employs a Generative Flow Network (GFlowNet) [Ben+21; Del+22; Mal+22] to fine-tune an LLM, based on my joint work with collaborators at Mila [Hu+24]. GFlowNets are trained to learn a sampling policy P_F over trajectories (e.g., sequences of reasoning steps and code tokens in our case) such that the probability of sampling a trajectory terminating with a program p is proportional to an unnormalized reward $R(p)$, i.e., $P_F(p) \propto R(p)$ [Ben+21]. This contrasts with traditional reinforcement learning (RL), which optimizes for the single best program ($\arg\max_p R(p)$).

Specifically, SynthStats fine-tunes an LLM using a GFlowNet objective: the Sub-Trajectory Balance (SubTB) loss [Hu+24; Mad+23], to perform *amortized sampling* over chains-of-thought Z (CoTs) leading to PPL programs with the desired properties. The LLM generates a sequence including reasoning steps (in `<thinking>` tags) followed by the PPL code. The reward $R(p)$ reflects program quality based on execution, statistical validity, and alignment with the prompt. The LLM-specific SubTB loss for a sequence $Z = z_{1:n}\top$ (where $\top$ denotes termination) is:

$$\mathcal{L}(Z;\theta) = \sum_{0\le i<j\le n} \left(\log \frac{R(z_{1:i}\top)\prod_{k=i+1}^{j} q_{\text{GFN}}(z_k|z_{1:k-1}) q_{\text{GFN}}(\top|z_{1:j})}{R(z_{1:j}\top) q_{\text{GFN}}(\top|z_{1:i})} \right)^2 \tag{1}$$

where $R(z_{1:k}\top)$ is the reward for terminating after k tokens, and q_{GFN} represents the LLM's forward sampling probabilities (including termination $\top$). Minimizing this loss enforces flow consistency, driving the policy towards the target distribution $p(Z) \propto R(Z)$. By sampling proportionally to the reward, GFlowNets naturally handle multi-modality of the posterior (multiple good programs) and capture uncertainty, providing a diverse set of candidate models rather than a single, potentially overconfident solution. This distribution-matching objective makes the approach more robust to reward misspecification compared to reward maximization in RL [Dee+25; PBS21; Pan+23].

While the semantic work in Part I does not directly dictate the specifics of the probabilistic program synthesis pipeline of Part II, the principles derived (e.g., semantic soundness, compositionality, principled uncertainty handling via PPLs) inform the *desiderata* for the synthesized programs and the design of the framework. The goal is to leverage the generative capabilities of LLMs, while grounding the output in the formal structure and desirable properties of PPLs, leading to easier generation of interpretable statistical models.

Core Research Questions

This work addresses three core questions:

Q1: *Semantic Foundations*: What categorical structures (like QBS or presheaf models) are suited for modeling higher-order Bayesian nonparametrics [Das+23], and how to support stochastic memoization while preserving key properties of categorical probability like the dataflow property [KS23]?

Q2: *Practical Realisation*: How can these semantic structures be implemented in a PPL like LazyPPL [Laz25], and what inference algorithms (e.g., Metropolis-Hastings-Green on infinite rose trees [Das+23]) are compatible with laziness and unbounded state spaces? Further, how to make this efficient in practice with GPU-accelerated inference, and how to support expressive priors like state-of-the-art transformer models?

Q3: *LLM-Guided Synthesis*: Can GFlowNet-fine-tuned LLMs [Hu+24] learn to sample distributions over PPL programs that are statistically and semantically sound, following principles from Part I, and aligned with user specifications, thereby advancing techniques for safer automated statistical model discovery?

Thesis Roadmap and Structure

The structure of the planned thesis follows the organization illustrated in Figure 2. Part I will detail the theoretical PPL foundations. Part II will introduce and develop the SynthStats framework, outlining its applications and planned experimental validation. The anticipated timeline for completion is discussed below.

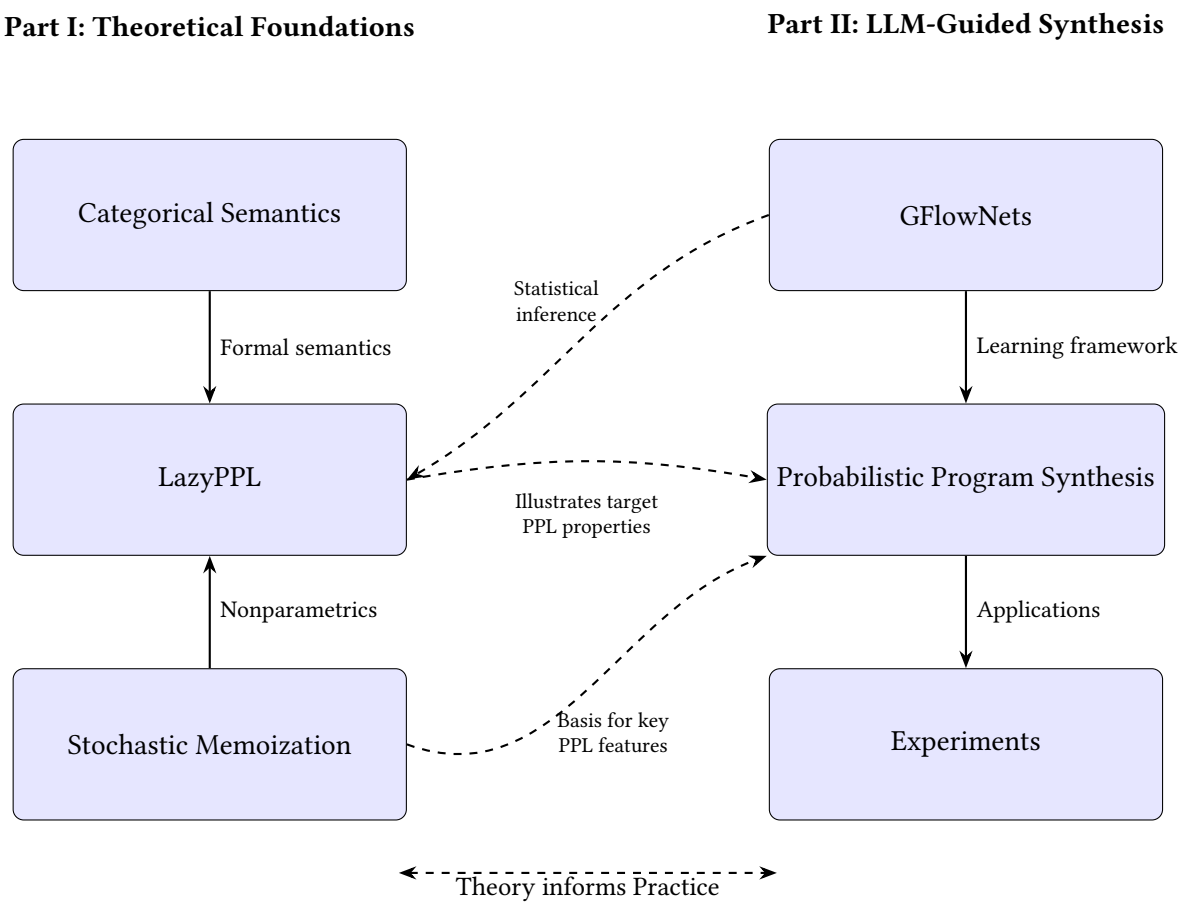

Figure 2: Planned thesis roadmap. Solid arrows denote primary dependencies within each part; dashed arrows indicate how theoretical principles from Part I inform the goals and design of the synthesis work in Part II, and vice versa.

Timeline and Future Directions

This subsection summarises the remaining work required to complete the DPhil by October 2025, and briefly mentions research avenues extending beyond the thesis. The plan, accounting for work completed up to the start of Q2 2025 (April), is shown in Figure 3. Key technical milestones are aligned with the thesis chapters and experimental plan outlined previously.

For the period Q2 2025 - Q4 2025 the priorities are organised around three work-packages:

LazyPPL JAX Backend. Re-implement the core monads Prob and Meas with a JAX backend for GPU acceleration (potentially enabling GFlowNet-based statistical inference within the PPL) and support for state-of-the-art transformers as priors. Expose a compatible API with the Haskell version, and benchmark performance against the original Haskell backend.

SynthStats Prototype. Finalise the GFlowNet-LLM pipeline for generating PyMC programs from natural-language prompts, via posterior amortized inference over chains-of-thought with GFlowNets. Implement the multi-component reward function. Build a benchmarking suite for automated model discovery and conduct experiments against baseline methods.

Thesis Writing. Complete remaining chapters, crafting a coherent narrative that links the theoretical foundations of Part I with the probabilistic program synthesis framework of Part II. Circulate full draft to my supervisor by September 2025.

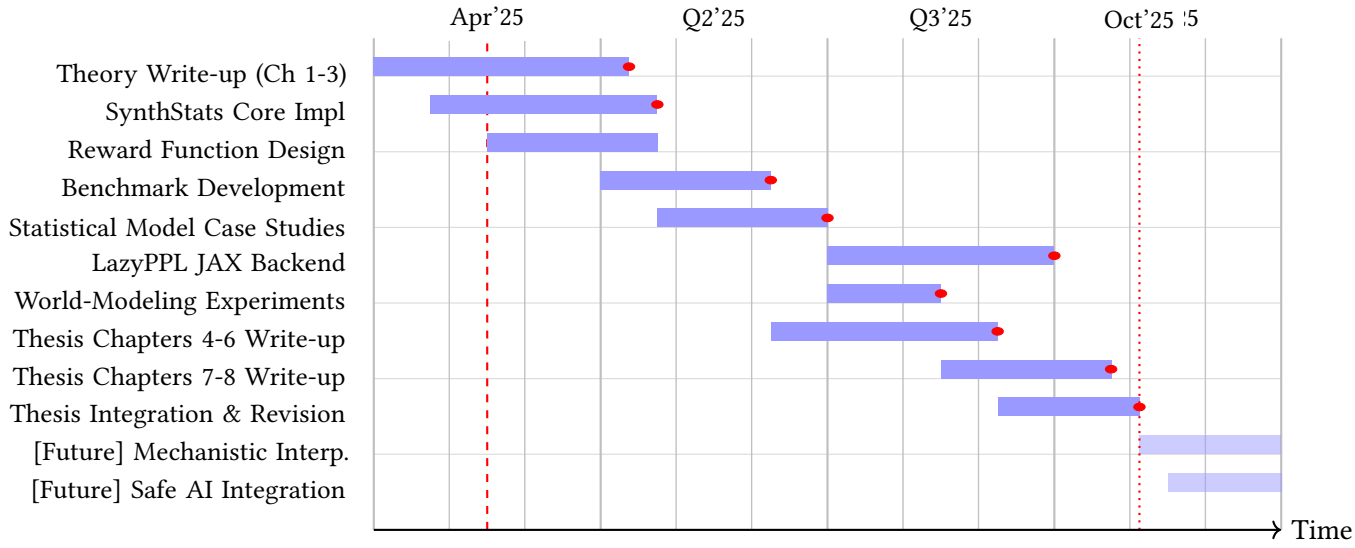

Figure 3: Overview of the DPhil research timeline (updated as of April 2025). Red dots mark target completion points. The dashed line indicates the current date (April 2025), and the dotted line marks the thesis deadline (October 2025).

Concluding Remarks

The work presented in this overview outlines a path from foundational categorical semantics of probabilistic programming (Part I) to a practical probabilistic program synthesis framework (SynthStats, Part II). By delivering the JAX backend for LazyPPL and the SynthStats prototype within the stated timeline, my goal is to combine semantic principles and state-of-the-art deep learning models for more interpretable and uncertainty-aware AI systems. The longer-term agenda aims for this research to have a meaningful impact in automated model discovery, AI safety, and mechanistic interpretability.

References

[Ack+24] Nate Ackerman et al. "Probabilistic Programming Interfaces for Random Graphs: Markov Categories, Graphons, and Nominal Sets". In: *Proc. ACM Program. Lang.* 8.POPL (2024), 61:1–61:31. DOI: 10.1145/3632903.

[Ant25] Anthropic. *Claude 3.7 with extended thinking*. https://www.anthropic.com/news/visible-extended-thinking. 2025.

[Ben+21] Yoshua Bengio et al. "GFlowNet Foundations". In: (2021). arXiv: 2111.09266 [cs.LG].

[Das+23] Swaraj Dash et al. "Affine Monads and Lazy Structures for Bayesian Programming". In: *Proc. ACM Program. Lang.* 7.POPL (2023), pages 1338–1368. DOI: 10.1145/3571239.

[Dee+25] DeepSeek-AI et al. "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning". In: *arXiv preprint arXiv:2501.12948* (2025).

[Del+22] Tristan Deleu et al. "Bayesian Structure Learning with Generative Flow Networks". In: *arXiv preprint arXiv:2202.13903* (2022). Revised version (v2), submitted on 28 Feb 2022 and revised on 28 Jun 2022. DOI: 10.48550/arXiv.2202.13903v2.

[Goo25] Google. *Gemini API: Thinking with Gemini*. https://ai.google.dev/gemini-api/docs/thinking. 2025.

[Gra+24] Aaron Grattafiori et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.

[Hu+24] Edward J. Hu et al. "Amortizing intractable inference in large language models". In: (2024).

[KS23] Younesse Kaddar and Sam Staton. "A model of stochastic memoization and name generation in probabilistic programming: categorical semantics via monads on presheaf categories". In: *Electronic Notes in Theoretical Informatics and Computer Science* Volume 3 - Proceedings of MFPS XXXIX, 10 (2023). ISSN: 2969-2431. DOI: 10.46298/entics.12291. URL: https://entics.episciences.org/12291.

[Kah12] Daniel Kahneman. *Thinking, fast and slow*. English. London: Penguin, 2012. ISBN: 9780141033570 0141033576.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN: 0262013193.

[Laz25] LazyPPL Team. *LazyPPL: A Haskell Library for Bayesian Probabilistic Programming*. LazyPPL website. Accessed: 2025-02-06. 2025. URL: https://lazyppl-team.github.io/.

[Mad+23] Kanika Madan et al. "Learning GFlowNets from Partial Episodes for Improved Convergence and Stability". In: *Proceedings of the 40th International Conference on Machine Learning (ICML)*. Volume 202. Proceedings of Machine Learning Research. PMLR, 2023, pages 23467–23483. URL: https://arxiv.org/abs/2209.12782.

[Mal+22] Nikolay Malkin et al. "Trajectory balance: Improved credit assignment in GFlowNets". In: *arXiv preprint arXiv:2201.13259* (2022). Submitted on 31 Jan 2022, revised v3 on 4 Oct 2023; NeurIPS 2022.

[Mee+18] Jan-Willem van de Meent et al. "An Introduction to Probabilistic Programming". 2018. URL: https://arxiv.org/abs/1809.10756.

[Ope+24] OpenAI et al. *OpenAI o1 System Card*. 2024. arXiv: 2412.16720 [cs.AI]. URL: https://arxiv.org/abs/2412.16720.

[PBS21] Alexander Pan, Kush Bhatia and Jacob Steinhardt. "The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models". In: *International Conference on Learning Representations*. 2021.

[Pan+23] Richard Yuanzhe Pang et al. "Reward Gaming in Conditional Text Generation". In: *arXiv preprint arXiv:2211.08714* (2023).

[Sta+17] Sam Staton et al. "Commutative Semantics for Probabilistic Programming". In: Lecture Notes in Computer Science 10201 (2017), pages 855–879. DOI: 10.1007/978-3-662-54434-1_32.

[Ste21] Dario Maximilian Stein. "Structural Foundations for Probabilistic Programming Languages". University of Oxford, 2021. 221 pages.

[VKS19] Matthijs Vákár, Ohad Kammar and Sam Staton. "A Domain Theory for Statistical Probabilistic Programming". In: *Proc. ACM Program. Lang.* 3.POPL (2019), 36:1–36:35. DOI: 10.1145/3290349.

[Wei+22] Jason Wei et al. "Chain-of-thought prompting elicits reasoning in large language models". In: *Advances in neural information processing systems* 35 (2022), pages 24824–24837.

[Woo+09] Frank Wood et al. "A Stochastic Memoizer for Sequence Data". In: (2009), pages 1–8. DOI: 10/fg8z4q. URL: http://portal.acm.org/citation.cfm?doid=1553374.1553518 (visited on 01/21/2022).