Exploiting Constraint Decomposition In Probabilistic Neuro-Symbolic Layers

Leander Kurscheidt[1] Antonio Vergari[1]

[1]School of Informatics, University of Edinburgh, UK

Abstract

Computing exact probabilistic predictions under algebraic constraints is a notoriously hard task in high-dimensions. In this paper, we focus on a number of application scenarios, such as trajectory prediction, in which a *high-dimensional* distribution (possibly parameterized by a neural network) needs to be exactly integrated over a *lower-dimensional* algebraic constraint. We can exploit this problem structure in a framework like `PAL`, where probability distributions and constraints are modeled independently, greatly speeding up the computation of the constraint probability. Our solution, named `PcPAL`, can capture intricate dependencies over variables while delivering guaranteed satisfaction of the constraints while retaining tractability, as shown on trajectory prediction tasks.

1 INTRODUCTION

For AI systems operating in *safety-critical* settings, reliable decision-making depends on two capabilities: modeling uncertainty through predicting *probabilities* and *enforcing the hard constraints* dictated by the environment. Probabilistic neuro-symbolic (NeSy) methods [Garcez et al., 2019, 2022, De Raedt et al., 2021] promise to enable trustworthy AI by integrating symbolic reasoning over high-level constraints into neural predictors. While recent NeSy advances like `PAL` [Kurscheidt et al., 2025] enable computing probabilities over SMT algebraic constraints, scaling this computation to high-dimensions is still an open problem .

In this paper, we focus on scaling these probabilistic NeSy systems when it is necessary to guarantee reliable predictions over an extended time horizon rather than at a single instant [Liu and Hwang, 2010, Tang and Salakhutdinov, 2019]. In this scenarios, while the probability distribution needs to capture dependencies in high-dimensional spaces,

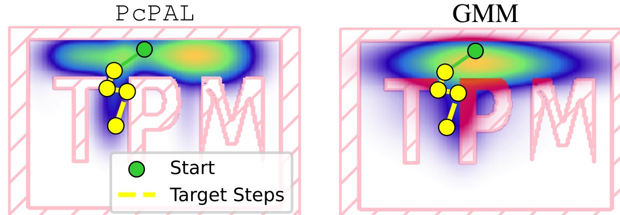

Figure 1: **`PcPAL` combines predictions over long horizons with guaranteed constraint-satisfaction**. We show the spatial distribution over future positions obtained by marginalizing out time for `PcPAL` and GMM predictions. Constraint violations are highlighted in red. See Section 5.2.

e.g., correlations between the time- and space-axis, the hard constraints are defined over a lower-dimensional space. For example, safety distance to other actors or boundaries imposed though e.g. building or obstacles are the same across all time steps [Zeng et al., 2021, Chen et al., 2025]. Focusing on problems with constraints that decompose into independent low-dimensional subgroups allows us to introduce `PcPAL`, which exploits the same formulation of `PAL`, but models the probabilistic component as a probabilistic circuit (PC) [Darwiche, 2003, Vergari et al., 2019, Choi et al., 2020]. This enables us to scale the computation of the probability of a constraint being satisfied, also known as the weighted model integral (WMI) , to longer horizons (Fig. 1). We do this by investigating the following research-questions: **R1:** How to scale WMI to high dimensions via time-decomposable constraints? **R2:** How to extend `PAL` (`PcPAL`) to high-dimensional, non-factorized predictions under decomposable constraints? **R3:** Does the proposed approach work in practice? We answer them next.

2 CONSTRAINED PROBABILISTIC PREDICTIONS

Notation. We use uppercase letters to denote random variables (e.g., X, Y) and lowercase letters for their realizations (e.g., x, y). Bold symbols represent (random) vectors (e.g., $\mathbf{X}, \mathbf{Y}$) and their joint assignments (e.g., $\mathbf{x}, \mathbf{y}$). Greek letters such as Φ, Δ, or Φ denote logical formulas mapping real-valued variables to Boolean values (false or true). An assignment $\mathbf{x}$ is said to satisfy a constraint Φ, written as $\mathbf{x} \models \Phi$, if substituting $\mathbf{x}$ into Φ evaluates it to true. The corresponding indicator function $\mathbb{1}\{\mathbf{x} \models \Phi\}$ equals 1 if $\mathbf{x}$ satisfies Φ, and 0 otherwise.

SMT($\mathcal{LRA}$) Constraints. In order to model the constraints imposed by the environment, we turn to SMT($\mathcal{LRA}$), or satisfiability modulo theory over linear real arithmetic (Barrett et al. 2021). SMT($\mathcal{LRA}$) is a logic over both real-valued and boolean variables, and represents constraints using piecewise-linear boundaries. We consider quantifier-free SMT formulas over continuous variables $\mathbf{X}$, consisting of linear (in)equalities of the form $\sum_i a_i X_i \bowtie b$, where $\bowtie \in \{\leq, =\}$, combined using Boolean connectives, so conjunctions ($\wedge$), disjunctions ($\vee$), and negations ($\neg$).

Example 1 (SMT formula)**.** *Consider the following SMT formula over the bivariate variables* $\mathbf{X}^{(1)} = \{X_1^{(1)}, X_2^{(1)}\}, \mathbf{X}^{(2)}, \dots, \mathbf{X}^{(n)} = \{X_1^{(n)}, X_2^{(n)}\}$, *which defines polytope-constraint per time-step t:*

$$\Delta_t(\mathbf{X}^{(t)}) = X_1^{(t)} \in [0,2] \wedge X_2^{(t)} \in [0,2] \wedge (X_1^{(t)} + X_2^{(t)} \leq 1)$$
$$\Delta(\mathbf{X}) = \bigwedge_{t=1}^{n} \Delta_t(\mathbf{X}^{(t)})$$

But furthermore, SMT($\mathcal{LRA}$) can scale to complex, non-convex constraints, as seen in figure 1.

Setting. Our task is to learn a parameterized conditional distribution of $\mathbf{Y}$ given $\mathbf{X}$, i.e. $p_{\boldsymbol{\Theta}}(\mathbf{Y} \mid \mathbf{X})$ from a dataset $\mathcal{D} = (\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ consisting of realizations of the continuous random variables $\mathbf{X}$ and $\mathbf{Y}$. In contrast to the classical unconstrained setting, the support of $\mathbf{Y}$ is restricted by a formula $\Phi \in$ SMT($\mathcal{LRA}$), ensuring that $p_{\boldsymbol{\Theta}}(\mathbf{Y} \models \neg\Phi \mid \cdot) = 0$. We further assume that Φ factorizes into independent low-dimensional constraints

$$\Phi(\mathbf{Y}) = \bigwedge_i \Phi_i(\mathbf{Y}_{\text{group}_i}) \qquad \text{(Constraint Factorization)}$$

with the variable scopes group_i being pairwise non-overlapping and $|\mathbf{Y}_{\text{group}_i}| \ll |\mathbf{Y}|$. The formula in Example 1 satisfies this desiderata.

`PAL` tackles the problem of predicting a probability distribution over $\mathbf{Y}$ subject to constraints Φ, without exploiting any specific structure in Φ. It represents the unconstrained distribution over $\mathbf{Y}$ using a piecewise polynomial density q with parameters $\boldsymbol{\lambda}$, which are predicted by a neural network $f_{\boldsymbol{\psi}}$. The constrained distribution is then obtained by restricting q to the feasible set and renormalizing via

$$p_{\boldsymbol{\Theta}}(\mathbf{Y} \mid \mathbf{x}) = \frac{q(\mathbf{Y}; \boldsymbol{\lambda}{=}f_{\boldsymbol{\psi}}(\mathbf{x}))\mathbb{1}\{\mathbf{Y} \models \phi\}}{\int q(\mathbf{y}'; \boldsymbol{\lambda}{=}f_{\boldsymbol{\psi}}(\mathbf{x}))\mathbb{1}\{\mathbf{y}' \models \phi\}\, d\mathbf{y}'} \qquad \text{(PAL)}$$

It is important to notice that the normalizing factor is not a constant but a function of $\mathbf{x}$, or, if we change perspective, a function of $\boldsymbol{\lambda}$. Efficiently and computing it is a major challenge. `PAL` approaches the problem by considering the symbolic computation in $\boldsymbol{\lambda}$, where $\boldsymbol{\lambda}$ is a simple parametrization via the coefficients of the monomials:

$$I(\boldsymbol{\lambda}) = \int q(\mathbf{y}'; \boldsymbol{\lambda})\mathbb{1}\{\mathbf{y}' \models \phi\}\, d\mathbf{y}' \qquad \text{(PAL WMI)}$$
$$= \int \sum_\alpha \lambda_\alpha \prod_i (y_i')^{\alpha_i} \mathbb{1}\{\mathbf{y}' \models \phi\}\, d\mathbf{y}' \qquad (1)$$
$$= \sum_\alpha \lambda_\alpha \underbrace{\int \prod_i (y_i')^{\alpha_i} \mathbb{1}\{\mathbf{y}' \models \phi\}\, d\mathbf{y}'}_{:=M_\alpha(\phi),\text{ independent of }\boldsymbol{\lambda}} \qquad (2)$$

So by precomputing the constrained moments $M_\alpha(\phi)$, we can use knowledge compilation to turn PAL into an efficient, exact and differentiable operation. Computing $M_\alpha(\phi)$ is done through `GASP!`, which is a GPU-accelerated integration-backend that works through enumerating the feasible area and numerically integrating the polynomial over it without approximation. Unfortunately, both the number of monomials and the computational complexity of $M_\alpha(\phi)$ explodes in high dimensions, limiting `PAL` to problems with lower-dimensional predictions.

Probabilistic Circuits. PCs [Darwiche, 2003, Vergari et al., 2019, Choi et al., 2020] provide a framework for constructing expressive distributions by combining simple, typically univariate, distributions while supporting tractable operations such as conditioning and marginalization under suitable structural conditions [Vergari et al., 2021]. We define a simple class of tensorized PCs [Loconte et al., 2025]. We consider univariate distributions as base input distributions. Each variable X_i is mapped into a vector f_{X_i}:

$$f_{X_i} : \mathbb{R} \to \mathbb{R}^o, \qquad \text{with } f_{X_i}(X_i) = [p_j(X_i)]_{j=1}^o \qquad (3)$$

where p_j is a density of a univariate distribution that supports tractable computation of the CDF. Rather than concatenating these representations, we treat them as separate vectors throughout the circuit. Each vector $\mathbf{l}$ is associated with a scope $\text{sc}(\mathbf{l})$, indicating the subset of variables on which it depends. Acting on these vectors, we use three types of layers [Loconte et al., 2025]: the Kronecker-layer (K) that combines two layers using an outer product, the sum-layer (S) to transform a latent linearly, and the Tucker-layer (T) to combine two layers using a multilinear transformation:

$$\begin{aligned} &\mathrm{K} : (\mathbb{R}^a, \mathbb{R}^b) \to \mathbb{R}^{a \cdot b} && \mathrm{K}(\mathbf{l}, \mathbf{l}') = \text{vec}(\mathbf{l}\mathbf{l}'^\top) \\ &\mathrm{S} : \mathbb{R}^a \to \mathbb{R}^o && \mathrm{S}(\mathbf{l}) = \mathbf{W}\mathbf{l} \\ &\mathrm{T} : (\mathbb{R}^a, \mathbb{R}^b) \to \mathbb{R}^o && [\mathrm{T}(\mathbf{l}, \mathbf{l}')]_j = \mathbf{l}^T \mathbf{W}_j \mathbf{l}' \end{aligned}$$

where we assume that the layers $\mathbf{l}$ and $\mathbf{l}'$ are defined over disjoint sets of variables, i.e., the PC is *decomposable*. We can combine these layers until we reach a scalar which is the output of the PC. Computing the normalizing constant over such a circuit is now straightforward, the multi-linear nature of K, S and T enables us to push the actual integral from the overall computation back to the input-functions and therefore results in the same circuit but starting with the value of the integral over f_{X_i} instead of the density.

3 RELATED WORK

Many probabilistic NeSy methods [Marra et al., 2024] enforce constraints only in expectation via loss penalties, using propositional [Xu et al., 2018, Ahmed et al., 2023], fuzzy [Diligenti et al., 2017], physical [Stewart and Ermon, 2017], or algebraic [Fischer et al., 2019] formulations. This typically yields no feasible sampling procedure and does not guarantee constraint satisfaction. Unlike `PcPAL`, these approaches therefore do not provide valid constrained distributions. Neuro-symbolic approaches to sequential problems, such as time-series prediction, have so far received limited attention. A closely related line of work is relational neuro-symbolic Markov models (NeSy-MMs) [Smet et al., 2025]. In contrast to our setting, NeSy-MMs focus on hidden Markov models with constraints imposed on the latent state space.While being able to express both discrete and continuous constraints using DeepSeaProblog [Smet et al., 2023], in contrast to `PcPAL`, closed form constrained distributions are not available and inference relies on approximate NeSy particle filtering.

4 EXPLOITING DECOMPOSABILITY

In this section, we scale `PAL` to high-dimensional problems by exploiting the constraint decomposition in (Constraint Factorization). Crucially, we decouple the factorization of the constraints from the structure of the predictive distribution: although the constraints factorize over non-overlapping low-dimensional groups, we construct an expressive distribution that allows dependencies across groups, while exploiting the structure for efficient computation of the normalization integral.

4.1 SCALING UP GASP!

In this section, we aim to answer **R1**, and establish whether we can exploit the decomposability to tractably scale the hard-constrained integral to high dimensions. As `PAL` is based on polynomial densities, we consider the task of integrating polynomials over SMT($\mathcal{LRA}$)-constrained domains. In general, this task is intractable, even for a bounded-degree polynomial over a single polytope of varying dimensionality [Baldoni et al., 2011]. Fortunately, for the WMI over decomposable formulas (Constraint Factorization), we can tractably compute the WMI. Under the assumption that the dimensionality of the subgroups is bounded ($\max_i |\mathbf{Y}_{\text{group}_i}| \leq d$), that AllSAT over each group $\Phi(\mathbf{Y}_{\text{group}_i})$ can be performed in polynomial time and that our polynomials decomposes (using addition/multiplication) according to the subgroups, we can compute the WMI tractably in $|\mathbf{Y}|$ (See 7).We therefore positively answer **R1** and recognize that we can exploit the decomposability to tractably compute the WMI-integral, although with the requirement that the polynomial, and therefore the density, must factorize accordingly.

4.2 SCALING UP PAL

We now want to build upon out results of **R1** to tackle **R2** without collapsing back to a simple factorization of the predictive distribution. We start by considering how the computation of the integral decomposes over the probabilistic circuit. By pushing down the integral in both K and T, we split the integral into lower-dimensional integrals over both $\mathrm{sc}(\mathbf{l})$ and $\mathrm{sc}(\mathbf{l}')$. For the WMI over a PC under Constraint Factorization, we can do the same as long as the scopes in the formulas decompose accordingly. For WMI over a PC under (Constraint Factorization), the same applies provided that the scopes of the constraints decompose accordingly. Using this observation, we can leverage **R1** by structuring the probabilistic circuits to respect the decomposition in (Constraint Factorization), while allowing arbitrary structure for the sub-circuits within each $\mathbf{Y}_{\text{group}_i}$. We call the use of probabilistic circuits to exploit the decomposable structure `PcPAL`. If we now require the input-functions to be polynomial input-distributions, the sub-circuits over $\mathbf{Y}_{\text{group}_i}$ are just linear combinations of those and still polynomial. Therefore, we can leverage `GASP!` and apply the `PAL` approach: We can pre-compute the constrained moments $M_\alpha(\Phi_i)$ over the sub-circuits and just like in `PAL` exactly and efficiently compute the integral over the lower-dimensional sub-circuits and combine these integrals according overall circuit to compute the partition function. In practice, we choose squared splines as the density of the univariate input-distribution and after predicting the parameters normalize them by their unconstrained integral. Additionally each row $\mathbf{W}_{r,\cdot}$ in the sum-layer S and $\mathrm{vec}(\mathbf{W}_i)$ for the tucker-layer T lie on the simplex, which can be achieved through a simple softmax-reparameterization. Under these conditions, the resulting circuit represents a normalized probability distribution over the unconstrained domain. To obtain a distribution satisfying the constraints Φ, we then restrict this distribution to the feasible region and renormalize by its integral over Φ.

5 EXPERIMENTS

To answer **R3**, we now empirically investigate the performance of our proposed method. We first analyses the scalability of `GASP!` for decomposable problems. We then turn

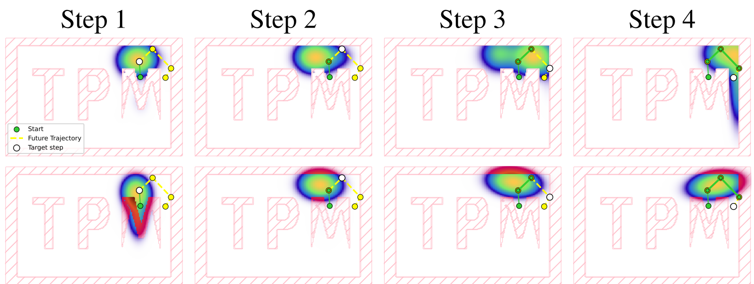

Figure 2: **PcPAL captures the tight correlations between consecutive steps, while natively wrapping probability mass around the infeasible obstacles.** We show step-wise conditional predictions for a 4-step rollout, where each step is conditioned on all previous steps and all the future steps are marginalized out. The top row is a `PcPAL`-prediction and the bottom row is a GMM.

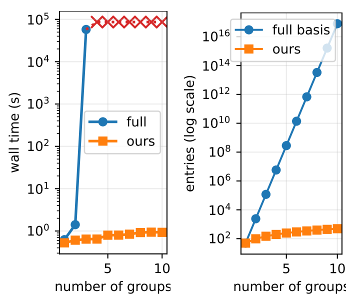

Figure 3: We can scale the WMI-integral when constraints decompose into independent subgroups. This is the case for prob. circuit PAL. Left: `GASP!` timings. Right: number of basis elements required.

Table 1: We compare held-out log-likelihood (LL), average- (ADE) and final displacement error (FDE), and the average probability of violating constraints $P(\neg\Phi)$ for `PcPAL`, GMM and IndPAL on TPM-Obstacles (See 9.3).

Model	LL	ADE	FDE	$P(\neg\Phi)$
`PcPAL`	**1.39**	**0.57**	**0.77**	**0.0**
GMM	0.94	0.59	0.79	0.35
IndPAL	-0.13	0.58	0.78	**0.0**

to a constrained trajectory prediction under obstacles.

5.1 SCALING UP GASP!

We first investigate the scalability of `GASP!` and compare the timings of computing all the required moments $M_\alpha(\Phi)$ for `PAL` vs. all the required moments $M_\alpha(\Phi_i)$ for `PcPAL`. We generate formulas of $2n$ dimensions with 2-dimensional subgroups. For each subgroup we generate 2 random, non-overlapping simplices. Due to squared splines being used in `PAL` and `PcPAL` to model densities, we prepare the moments for a 6-degree polynomial per dimension. Even just enumerating the required basis elements for `PAL` results in an combinatorial explosion, in addition to the intractable WMI for each constrained moment. This can be seen in figure 3, where we show the overall time to prepare the required moments and the number of basis elements (constrained moments) to integrate over. One can see that we can only prepare all the constrained moments up to 6 dimensions before timing out, whereas `PcPAL` scales linearly. We therefore conclude that, using `GASP!`, we can scale the required WMI over the constrained moments for `PcPAL` to high dimensional problems. Details on the hardware-setup are given in the appendix in 8.

5.2 CONSTRAINED TRAJECTORY PREDICTION

We evaluate `PcPAL` on a synthetic, constrained trajectory prediction task (TPM-Obstacles). The 4-step trajectories are constrained to the feasible area defined as the complement of the letters TPM, as seen in figure 1. For each input $\mathbf{x}$, we sample feasible trajectories $\mathbf{y}$ originating from $\mathbf{x}$ and extending outward in 8 directions (see 9.1 for more details). The constraints act per step. We generate 80.000 datapoints and train on 60.000 with 10.000 for validation and test. This task combines strong dependencies between future time steps with high uncertainty over the initial direction, requiring both accurate modeling of cross-time correlations and multi-modal predictive distributions. We compare `PcPAL` to a constraints-unaware gaussian mixture-model predicted by a neural network (GMM) [Bishop, 1994] and a prediction of 4 independent `PAL` distributions (IndPAL) also by a neural network conditioned on $\mathbf{x}$. See 9.2 for more details. As visualized in figure 1 and 2, `PcPAL` models both the high correlation as well as the multi-model nature of the task, while wrapping the probability mass around the constraints. This can be also seen in table 1, where `PcPAL` obtains the best held-out log-likelihood, average- and final displacement error with guaranteed constraint satisfaction.

6 CONCLUSION

We introduce `PcPAL` for non-factorizable predictive modeling under decomposable constraints. We demonstrate tractable WMI computation and accurate modeling of cross-group dependencies while ensuring constraint satisfaction, and validate the approach on synthetic data. For future work, we plan to scale `PcPAL` to higher dimensions and evaluate it on real-world benchmarks, such as constrained trajectory prediction on Stanford Drone Dataset (SDD) [Robicquet et al., 2016, Kurscheidt et al., 2025], extending its applicability beyond the synthetic setting.

References

Kareem Ahmed, Kai-Wei Chang, and Guy Van den Broeck. A pseudo-semantic loss for autoregressive models with logical constraints. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Velleda Baldoni, Nicole Berline, Jesus De Loera, Matthias Köppe, and Michèle Vergne. How to integrate a polynomial over a simplex. *Mathematics of Computation*, 80 (273):297–325, 2011.

Clark Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability Modulo Theories. In *Handbook of Satisfiability*, volume 336 of *FAIA*, pages 1267–1329. IOS Press, 2 edition, 2021. ISBN 978-1-64368-160-3 978-1-64368-161-0. doi: 10.3233/FAIA201017.

Christopher M Bishop. Mixture density networks. 1994.

Yi-Hsuan Chen, Shuo Liu, Wei Xiao, Calin Belta, and Michael Otte. Control barrier functions via minkowski operations for safe navigation among polytopic sets. In *2025 IEEE 64th Conference on Decision and Control (CDC)*, pages 4481–4488. IEEE, 2025.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. oct 2020. URL http://starai.cs.ucla.edu/papers/ProbCirc20.pdf.

Adnan Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM (JACM)*, 50: 280–305, 2003.

Luc De Raedt, Sebastijan Dumančić, Robin Manhaeve, and Giuseppe Marra. From statistical relational to neural-symbolic artificial intelligence. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4943–4950, 2021.

Michelangelo Diligenti, Marco Gori, and Claudio Saccà. Semantic-based regularization for learning and inference. *Artif. Intell.*, 244:143–165, 2017. doi: 10.1016/J.ARTINT.2015.08.011. URL https://doi.org/10.1016/j.artint.2015.08.011.

Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin T. Vechev. DL2: training and querying neural networks with logic. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1931–1941. PMLR, 2019. URL http://proceedings.mlr.press/v97/fischer19a.html.

A Garcez, M Gori, LC Lamb, L Serafini, M Spranger, and SN Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *Journal of Applied Logics*, 6(4): 611–632, 2019.

Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Luis C Lamb, Leo de Penning, BV Illuminoo, Hoifung Poon, and COPPE Gerson Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation. *Neuro-Symbolic Artificial Intelligence: The State of the Art*, 342:1, 2022.

Leander Kurscheidt, Paolo Morettin, Roberto Sebastiani, Andrea Passerini, and Antonio Vergari. A probabilistic neuro-symbolic layer for algebraic constraint satisfaction. In Silvia Chiappa and Sara Magliacane, editors, *Conference on Uncertainty in Artificial Intelligence, Rio Othon Palace, Rio de Janeiro, Brazil, 21-25 July 2025*, Proceedings of Machine Learning Research, pages 2431–2471. PMLR, 2025. URL https://proceedings.mlr.press/v286/kurscheidt25a.html.

Weiyi Liu and Inseok Hwang. Probabilistic 4d trajectory prediction and conflict detection for air traffic control. In *Proceedings of the 49th IEEE Conference on Decision and Control, CDC 2010, December 15-17, 2010, Atlanta, Georgia, USA*, pages 1183–1188. IEEE, 2010. doi: 10.1109/CDC.2010.5717681. URL https://doi.org/10.1109/CDC.2010.5717681.

Lorenzo Loconte, Antonio Mari, Gennaro Gala, Robert Peharz, Erik Quaeghebeur, Gennaro Vessio, Antonio Vergari, et al. What is the relationship between tensor factorizations and circuits (and how can we exploit it)? *TMLR*, 2025.

Giuseppe Marra, Sebastijan Dumancic, Robin Manhaeve, and Luc De Raedt. From statistical relational to neurosymbolic artificial intelligence: A survey. *Artif. Intell.*, 328:104062, 2024. doi: 10.1016/J.ARTINT.2023.104062. URL https://doi.org/10.1016/j.artint.2023.104062.

Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII*, volume 9912 of *Lecture Notes in Computer Science*, pages 549–565. Springer, 2016. doi: 10.1007/978-3-319-46484-8_33. URL https://doi.org/10.1007/978-3-319-46484-8_33.

Lennert De Smet, Pedro Zuidberg Dos Martires, Robin Manhaeve, Giuseppe Marra, Angelika Kimmig, and Luc De Raedt. Neural probabilistic logic programming in discrete-continuous domains. In Robin J. Evans and Ilya Shpitser, editors, *Uncertainty in Artificial Intelligence, UAI 2023, July 31 - 4 August 2023, Pittsburgh, PA, USA*, volume 216 of *Proceedings of Machine Learning Research*, pages 529–538. PMLR, 2023. URL https://proceedings.mlr.press/v216/de-smet23a.html.

Lennert De Smet, Gabriele Venturato, Luc De Raedt, and Giuseppe Marra. Relational neurosymbolic markov models. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 16181–16189. AAAI Press, 2025. doi: 10.1609/AAAI.V39I15.33777. URL https://doi.org/10.1609/aaai.v39i15.33777.

Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In Satinder Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 2576–2582. AAAI Press, 2017. doi: 10.1609/AAAI.V31I1.10934. URL https://doi.org/10.1609/aaai.v31i1.10934.

Yichuan Charlie Tang and Ruslan Salakhutdinov. Multiple futures prediction. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15398–15408, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/86a1fa88adb5c33bd7a68ac2f9f3f96b-Abstract.html.

Antonio Vergari, Nicola Di Mauro, and Guy Van den Broeck. Tractable probabilistic models: Representations, algorithms, learning, and applications. *Tutorial at the 35th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2019.

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 13189–13201, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/6e01383fd96a17ae51cc3e15447e7533-Abstract.html.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5498–5507. PMLR, 2018. URL http://proceedings.mlr.press/v80/xu18h.html.

Jun Zeng, Bike Zhang, and Koushil Sreenath. Safety-critical model predictive control with discrete-time control barrier function. In *2021 American control conference (ACC)*, pages 3882–3889. IEEE, 2021.

Günter M Ziegler. *Lectures on polytopes*, volume 152. Springer, 1993.

Supplementary Material

Leander Kurscheidt[1] **Antonio Vergari**[1]

[1]School of Informatics, University of Edinburgh, UK

7 TRACTABILITY WMI

Under the assumption that the polynomial decomposes (with addition/multiplication) along the groups of the constraints, by linearity of the integral, we can decompose the integral into the same factorization and therefore reduce it to the integral over the groups. As the dimensionality is bounded, we are looking at a WMI over a bounded dimension. Since we assume AllSAT is polynomial, we can enumerate the polytopes that make of the feasible space, of which we can only have at most polynomial many. As the description size of the polytope must itself be polynomial, the number of constraints defining the polytopes must be polynomial in size. Therefore, I can only have polynomially many simplices [Ziegler, 1993]. As integrating a polynomial over a simplex of bounded degree is a tractable operation [Baldoni et al., 2011], we have overall tractability of the weighted model-integral.

8 `GASP!` BENCHMARK

This benchmarks generate a varying number of two-dimensional groups connected via constraints. Each group is connected by generating two non-overlapping, two-dimensional simplices. We generate benchmark data from 1 to 10 groups. We compare the time it takes to compute all the constrained moments for a decomposable polynomial of degree 6 per dimension using `GASP!` for the enumeration of `PAL` (not exploiting decomposability) vs. `PcPAL` (which exploits the decomposability). We run the benchmark on a NVIDIA L40-GPU with a timeout of 24 hours per group.

9 TPM-OBSTACLES

9.1 DATASET

For the TPM-Obstacles, we generate 80.000 4-step trajectories around uniformly-distributed feasible starting-points inside our constraints (see 1 for a visualization of the constraints). Our proposal distribution for the constraints first samples one of 8 directions, and then samples steps along the line (step-size 0.22, points samples via a gaussian centered on the next-step position and with an standard-deviation of 0.20). This proposal distribution is visualized in figure 4. We reject infeasible trajectories. Samples of the constrained trajectories are visualized in figure 5

9.2 MODELS

For `PcPAL` we choose a hidden-Markov model-inspired architecture. We start with n input-functions each and then apply the CP-layer (pairwise product). Then, we have two options: **I.** (tucker) We have a sum-layer to the latent and combine two latents consecutively using a tucker layer, or **II.** (cp) we have skip the sum-layer, combine two latents again using a tucker-layer, but the final combination is cp-layer again. Importantly, density of the input-distribution is a squared spline, but we do not predict the parameters of the squared spline directly, but rather predict the parameters of gaussian and derive the

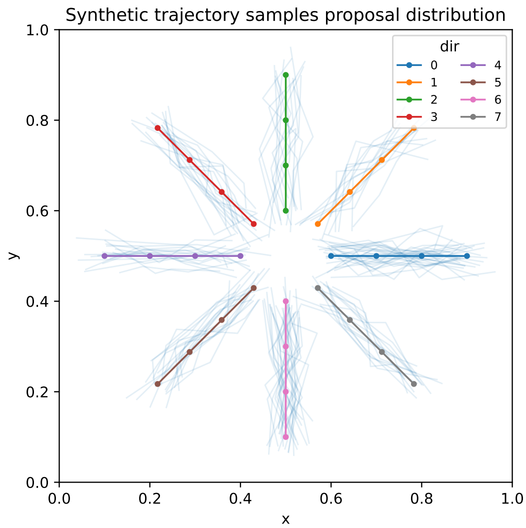

Figure 4: Proposal distributions for our trajectories, centered on $(0.5, 0.5)$

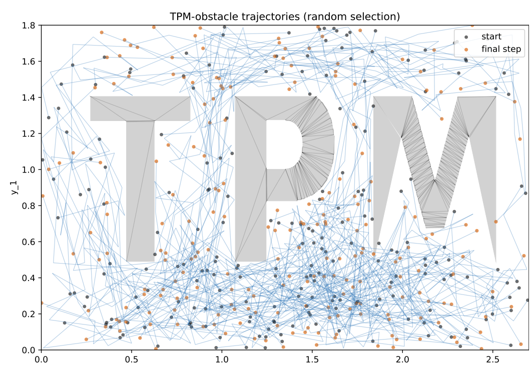

Figure 5: Randomly sampled trajectories of the TPM-Obstacles-dataset, infeasible area shown in grey.

parameters of the spline by evaluating the (root of) the gaussian density at the knot-point and taking the derivative of the (root of) the gaussian density at the knot points.

We run the experiment for 20 epochs using a learning rate of 0.0001 with Adam.

The parameters get predicted by a fully connected neural network with hidden-size 128. We perform a grid-search over the options:

- **type**: tucker and cp
- **number of knots**: 22, 26, 30, 34, 38
- **number of input-functions (per dim)**: 16, 20, 24, 28, 32

For the conditional GMM, we have two covariance parametrizations we predict with a neural network. Either full covariance (over all steps, so a 8-dimensional gaussian) or variances per step. We again choose a fully connected network but with a hidden dimension of 320 and run a grid-search over:

- **type** full and per-step covariance
- **numb. of mixtures** 8, 16, 24, 32, 48

For the independent PAL we directly predict the parameters of the spline using the fully-connected network. We have a fully-connected neural network with hidden dimension of 128. We run a grid-search over:

- **numb. of knots** 8, 12, 16, 22, 30

- **numb. of mixtures** 8, 12, 16, 24, 32

9.3 EVALUATION

We evaluate the models using:

- **LL** held-out log-likelihood on the test-set
- **ADE** (average displacement error) average RSME between a single sampled trajectory from the model and the ground-truth datapoint over all datapoints in the test-set. A common metric to asses the quality of generated trajectories.
- **FDE** (final displacement error) average RSME between the final step of a single sampled trajectory from the model and the ground-truth final step. A common metric to asses the final point where a generated trajectory might end up.
- **Average Constraints Violation** ($P(\neg\Phi)$: For models that do not guarantee constraint-violation, we sample 512 trajectories per datapoint in the test-set and approximate the probability of sampling a feasible trajectory, we then average the probability over the datapoints in the test-set.