Towards a Unified Framework for Probabilistic Verification of AI Systems

Paolo Morettin [a;*], **Andrea Passerini**[a] **and Roberto Sebastiani**[a]

[a]DISI, University of Trento

Abstract. The probabilistic formal verification (PFV) of modern AI systems is made particularly challenging by the widespread use of machine learning (ML) models. On the one hand, the techniques developed by the formal methods community are not suited to the verification of ML models. On the other hand, the contributions in the ML community so far have been limited to ad-hoc algorithms for specific classes of models and/or properties. First, we propose a unifying framework for the PFV of AI/ML systems, trying to frame the problem in the most general terms possible. Then, we show how to solve the quantitative verification task by reducing it to Weighted Model Integration (WMI) computations. We conclude by describing a number of open problems and research directions related to this promising approach.

1 Introduction and motivation

Given the increasing ubiquitousness of artificial intelligence (AI) in our everyday life, the development of verification techniques for modern AI systems is considered an important problem [48, 3].

In the last three decades, model checking has made huge progress in the verification of both hardware and software systems, guaranteeing the deployment of safe systems in many fields [17]. The most prominent approach in model checking reduces the verification problem to a sequence of decision or optimization problems.

This paradigm is also the mainstream approach in the verification of machine learning (ML) models, reducing the task to combinatorial reasoning problems like Satisfiability Modulo Theories (SMT) [28, 30, 31, 23, 19], linear (LP) [8] or mixed integer-linear programming (MILP) [53, 15]. This approach can answer whether a property is satisfied or not, but it falls short when we need to compute how likely a property will hold in an uncertain environment. The research community has identified many properties that are inherently quantitative, including fairness [22] and robustness [52].

Techniques developed in the field of *probabilistic formal verification* (PFV) [54, 18, 34, 29] have been applied to increasingly complex sequential stochastic systems, such as iterative probabilistic programs or network protocols. The scalability of model checking for sequential models with probabilities has been achieved by making strong distributional assumptions on the priors [26], such as independency or unimodality. For instance, random variables such as transition delays in timed automata are usually modelled with independent exponential distributions. In these scenarios, the high dimensionality of the search space is due to the sequential nature of the system under verification, rather than its inherent combinatorial complexity.

* Corresponding Author. Email: paolo.morettin@unitn.it.

Given the restrictive nature of the distributions considered, the techniques developed in this field do not readily support the verification of modern AI systems, which typically include ML components. In contrast with the traditional PFV use cases, these systems are highly structured parametric models, solving predictive problems over high-dimensional, multi-modal distributions.

The probabilistic verification on ML models is in its infancy. The large majority of works have focused on verifying whether a specific property, such as robustness [36, 56, 38] or fairness [9], holds with probability higher than a given threshold. While useful, these techniques provide little information on how far the system is from satisfying certain requirements. Alternatively, quantitative verification tasks can be reduced to a combinatorial counting problem. This approach was initially applied to the evaluation of loop-free probabilistic programs with univariate priors on random variables [16, 24]. This approach was later extended to fairness verification of decision procedures based on small ML models like decision trees, support vector machines and shallow piecewise-linear neural networks [2]. Quantitative robustness verification of binarized neural networks was also reduced to a counting problem, leveraging the advances in approximate counting over propositional logic theories [5].

In general, due to the large variety of models and their inherent combinatorial complexity, verification efforts in the ML community typically target specific classes of predictors, properties, or both. Investigating the probabilistic verification of AI systems under the lenses of a unified framework has several benefits. First, it will increase the chances that the progress made will generalize to the broadest possible context. In fact, on a smaller scale, developing a unified approach has proven to be beneficial to the qualitative robustness verification of piecewise neural networks [13]. Second, safety and trustworthiness are complex concepts that cannot be captured by a single evaluation metric or formal property. In our quest for more trustable AI systems, developing a single framework for the certification of many properties is a promising yet mostly unexplored research direction.

To fill this gap, we propose a unifying formulation for the probabilistic verification of AI systems. In the spirit of formalizing the problem in the most general terms possible, we consider the quantitative task rather than its more scalable decision variant. Computing how close a system is to satisfying a formal requirement is useful even when no distributional assumption is made, by uniformly distributing the uncertain variables over their domain [5]. An ideal framework should satisfy a number of desiderata, including:

D1) It should support *arbitrary distributions* over continuous and discrete domains, including complex models learned from data.

D2) It should provide a single representation language for arbitrary combinations of logical and algebraic constraints, with constructs that are flexible enough to *represent a broad range of AI systems*.

D3) The same representation language should be able to *encode many properties* of interest, ranging from ML-specific to system-wide properties that are typically considered in PFV.

In this paper, we show how a recent computational formalism, dubbed Weighted Model Integration (WMI) [10], provides a concrete implementation of a framework that matches our desiderata. Broadly speaking, WMI is the task of computing probabilities of arbitrary combinations of logical and algebraic constraints given a structured joint distribution over both continuous and discrete variables. Our contribution is twofold, we propose:

1. a unified perspective on the PFV of AI systems;
2. a reduction of the tasks defined above to WMI computations.

After providing the necessary background on WMI in Section 2, we present our two contributions in Section 3. We conclude in Section 4 by discussing the limitations of this work and and proposing a number of promising research directions.

2 Background on Weighted Model Integration

We assume the reader is familiar with the basic syntax, semantics, and results of propositional and first-order logic. We adopt the notation and definitions in [41] — including some terminology and concepts from Satisfiability Modulo Theories (SMT) — which we summarize below. We will restrict SMT to quantifier-free formulas over linear real arithmetic (SMT($\mathcal{LRA}$)), which consist in atomic propositions over some set $\mathbf{A}$ (aka *Boolean atoms*), linear (in)equalities ($\sum_i a_i x_i \bowtie b$) over a set of continuous variables $\mathbf{x}$ s.t. a_i, b are rational constants and $\bowtie \in \{\leq, \geq, >, <, =, \neq\}$ (aka $\mathcal{LRA}$ *atoms*), all of them combined with the standard Boolean operators $\{\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow\}$. The notions of literal, clause, partial and total truth assignment extend straightforwardly to SMT($\mathcal{LRA}$) atoms. We represent truth assignments μ as conjunctions of literals, s.t. positive and negative literals in μ mean that the atom is assigned to true and false respectively.

For brevity, we introduce *interval formulas* $(x \in [l, u]) \stackrel{\text{def}}{=} (l \leq x) \wedge (x \leq u)$ and *if-then-else terms* $ite(.;.;.)$ s.t. $t = ite(cond; e_1; e_2) \stackrel{\text{def}}{=} (cond \rightarrow t = e_1) \wedge (\neg cond \rightarrow t = e_2)$.

SMT is concerned with finding a truth assignment μ to the atoms of a formula Δ that both tautologically entails Δ ($\mu \models_{\mathbb{B}} \Delta$) and is $\mathcal{LRA}$-satisfiable. We denote with $\mathcal{TTA}(\Delta)$ the set of all the $\mathcal{T}$-satisfiable total truth assignments satisfying Δ. For instance, if $\Delta \stackrel{\text{def}}{=} (x < 0) \vee (x > 1)$, then $\mu \stackrel{\text{def}}{=} (x < 0) \wedge \mu(x > 1) \models_{\mathbb{B}} \Delta$ but it is not $\mathcal{LRA}$-satisfiable. Thus, the formula has only two $\mathcal{LRA}$-satisfiable total truth assignments: $\mathcal{TTA}(\Delta) = \{(x < 0) \wedge \neg(x > 1), \neg(x < 0) \wedge (x > 1)\}$. (Hereafter, we may drop "$\mathcal{LRA}$-satisfiable" when referring to assignments in $\mathcal{TTA}(\Delta)$ since this is clear from context.) Notice every $\mu \in \mathcal{TTA}$ is a convex subregion of Δ. We denote with $\mu^{\mathcal{LRA}}$ the portion on μ mapping $\mathcal{LRA}$-atoms to truth values.

Example 1 *The SMT($\mathcal{LRA}$) formula $\Delta \stackrel{def}{=} (x \in [0,1]) \wedge (y \in [0,1]) \wedge (A \rightarrow (x + y \leq 1))$ defines a region over $\mathbf{A} \stackrel{def}{=} \{A\}$ and $\mathbf{x} \stackrel{def}{=} \{x, y\}$, depicted in Figure 1 (left). It results in 3 (total) $\mathcal{LRA}$-satisfiable truth assignments (in each μ_i we omit $(x \in [0,1]) \wedge (y \in [0,1])$ for short):*

$$\mathcal{TTA}(\Delta) = \{\overbrace{A \wedge (x + y \leq 1)}^{\mu_1}, \overbrace{\neg A \wedge (x + y \leq 1)}^{\mu_2}, \overbrace{\neg A \wedge \neg(x + y \leq 1)}^{\mu_3}\}. \quad \diamond$$

We introduce non-negative weight functions $w : \mathbf{x} \cup \mathbf{A} \mapsto \mathbb{R}^+$, which intuitively defines a (possibly unnormalized) density function over $\mathbf{A} \cup \mathbf{x}$. $w(\mathbf{x}, \mathbf{A})$ is defined by: (1) a SMT($\mathcal{LRA}$) formula χ, called the *support* of w, outside of which w is 0; (2) a combination of real functions whose integral over $\mu^{\mathcal{LRA}}$ is computable, structured by means of nested if-then-elses with $\mathcal{LRA}$-conditions (we refer to [41] for a formal definition of $w(\mathbf{x}, \mathbf{A})$). When (2) is a DAG with polynomial leaves, as depicted in Fig. 1 (right), it is equivalent to the notion of *extended algebraic decision diagram* (XADD) introduced by Sanner et al.[47]. $w_{[\mu]}$ denotes w restricted to the truth values of μ. The **Weighted Model Integral** of a formula $\Delta(\mathbf{x}, \mathbf{A})$ and a weight function $w(\mathbf{x},\mathbf{A})$ over χ is defined as:

$$WMI(\Delta, w) \stackrel{\text{def}}{=} \sum_{\mu \in \mathcal{TTA}(\Delta \wedge \chi)} \int_{\mu^{\mathcal{LRA}}} w_{[\mu]}(\mathbf{x})\, d\mathbf{x} \tag{1}$$

WMI generalizes Weighted Model Counting (WMC) [46]. As with WMC, WMI can be interpreted as the total unnormalized probability mass of the pair $\langle \Delta, w \rangle$, i.e. its *partition function*. The normalized conditional probability of Γ given Δ can then be computed as:

$$P(\Gamma | \Delta) = WMI(\Gamma \wedge \Delta, w) / WMI(\Delta, w). \tag{2}$$

Example 2 *Consider the formula Δ in Ex.1, with weighted literals:*

$$w_{(x+y\leq 1)}(x, y) = x, \qquad w_{\neg A}(x, y) = y,$$
$$w_\ell(x, y) = 1 \qquad \forall \ell \notin \{(x + y \leq 1), \neg A\}$$

That is, w defines the following piecewise density over the convex subregions of Δ (Fig. 1, center):

$$w(\mathbf{x}, \mathbf{A}) \stackrel{def}{=} ite((x + y \leq 1); x; 1) \times ite(A; 1; y)$$

with $\chi \stackrel{def}{=} \top$, so that: $\left\{ \begin{array}{ll} w_{[\mu_1]}(x, y) & = x \times 1 = x \\ w_{[\mu_2]}(x, y) & = x \times y = xy \\ w_{[\mu_3]}(x, y) & = 1 \times y = y \end{array} \right\}$.

Then, the weighted model integral of the pair $\langle \Delta, w \rangle$ is:

$$WMI(\Delta, w) = \int_0^1 \int_0^{(1-y)} x\, dx\, dy + \int_0^1 \int_0^{(1-y)} xy\, dx\, dy + \int_0^1 \int_{(1-y)}^1 y\, dx\, dy = \frac{1}{6} + \frac{1}{24} + \frac{1}{3} = \frac{13}{24}.$$

We can now compute normalized probabilities such as

$$P(A | \Delta) = \frac{WMI(A \wedge \Delta, w)}{WMI(\Delta, w)} = \frac{1/6}{13/24} = \frac{4}{13} \quad \diamond$$

Crucially for our purposes, Γ and Δ in Eq. 2 can be *arbitrarily complex* SMT-LRA formulas. This gives as a very powerful computational tool for quantifying the probability associated to complex events or properties.

This flexibility comes at a cost: being a strict generalization of model counting (#SAT), enumerating $\mathcal{TTA}$ is #P-hard. Moreover, integrating a density inside an arbitrary convex polytope is at least as hard as computing its volume, which is #P-hard in itself [6].

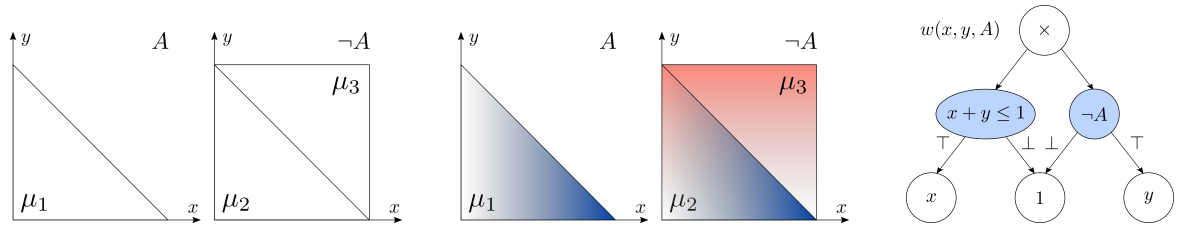

Figure 1. (Left) The hybrid region defined in Ex. 1. (Center) The weighted formula in Ex. 2. (Right) An equivalent XADD representation of the weight.

Nonetheless, since its introduction the problem has attracted the attention of many researchers, which resulted in steady advances in solving the problem [41, 32, 51], finding practical approximations [11, 20, 1] and identifying tractable subclasses [58]. We refer to [42] for a survey on the algorithmic progress in this field, focusing in this work on the opportunities offered by the WMI framework in the context of probabilistic formal verification of AI systems.

3 WMI-based verification of AI systems

In this Section we introduce our unifying perspective on the probabilistic verification of AI systems. We envision two roles, the *developer* of the system under verification $\mathcal{S}$ and the *verifier* (e.g. a regulatory body) that provides a specification of the requirements $\mathcal{R}$.

The developer is responsible of faithfully modelling $\mathcal{S}$ with a logical encoding $\Delta_{\mathcal{S}}$ of its deterministic behaviour, such as the functional relationships between its input and output. In many practical scenarios, however, $\mathcal{S}$ is not fully deterministic. For instance, the images captured by a camera might be subject to noise introduced by the sensor. The developer is then expected to model the uncertainty of $\mathcal{S}$ with a probabilistic model $P_{\mathcal{S}}$.

Similarly, the requirements $\mathcal{R}$ provided by the verifier include a logical formula $\Delta_{\mathcal{R}}$, encoding a desirable property that the system should satisfy with high probability. This probability is computed with respect to a probabilistic model that encodes the uncertainty of the environment where $\mathcal{S}$ is expected to be deployed, denoted by $P_{\mathcal{R}}$. A standard underlying assumption in verification tasks is that the formal specification of the requirement $\mathcal{R}$ is correct. Not only the property encoded in $\Delta_{\mathcal{R}}$ should not contain errors, but the model $P_{\mathcal{R}}$ should accurately model the environment.

The goal of the verification is computing the probability of the satisfaction of a requirement $\mathcal{R}$ by the system $\mathcal{S}$ given (1) a probabilistic model that jointly represents the uncertainty of $\mathcal{S}$ and of its surrounding environment and (2) a logical encoding of $\mathcal{S}$ and of the property described in $\mathcal{R}$ (Figure 2). In what follows, we denote with $\mathbf{x}$ and $\mathbf{y}$ the input and output of $\mathcal{S}$ respectively, and with $\mathbf{e}$, we denote extra environmental variables that are not observed by $\mathcal{S}$ but might be part of the specification, such as protected attributes that shouldn't be accessed by a fair hiring system (e.g. the gender of a job applicant). The probabilistic verification task can be formalized as computing :

$$P(\mathcal{R} \mid \mathcal{S}) = \frac{\int_{\Delta_{\mathcal{R}}(\mathbf{e},\mathbf{x},\mathbf{y}) \wedge \Delta_{\mathcal{S}}(\mathbf{x},\mathbf{y})} P(\mathbf{e},\mathbf{x},\mathbf{y})\, d\mathbf{e}\, d\mathbf{x}\, d\mathbf{y}}{\int_{\Delta_{\mathcal{S}}(\mathbf{x},\mathbf{y})} P(\mathbf{e},\mathbf{x},\mathbf{y})\, d\mathbf{e}\, d\mathbf{x}\, d\mathbf{y}}, \quad (3)$$

where $P(\mathbf{e},\mathbf{x},\mathbf{y})$ models the probabilistic relationship between inputs, outputs and environment, and factorizes as:

$$P(\mathbf{e},\mathbf{x},\mathbf{y}) = P_{\mathcal{S}}(\mathbf{y} \mid \mathbf{e},\mathbf{x}) \cdot P_{\mathcal{R}}(\mathbf{e},\mathbf{x}).$$

The definition of $P_{\mathcal{S}}(\mathbf{y} \mid \mathbf{e},\mathbf{x})$ accommodates both uncontrollable sources of uncertainty as well as any deliberate probabilistic mechanism in $\mathcal{S}$, even those that are not conditioned on the input, such as randomly sampled latent variables in a generative model.

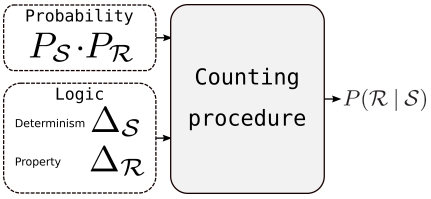

Figure 2. An overview of the PFV task. A probabilistic model encoding both the uncertainty of the system and the environment is complemented with a logical encoding of the determinism in $\mathcal{S}$ and a property $\mathcal{R}$. The output is the probability mass of $\mathcal{R}$ given $\mathcal{S}$.

Example 3 *Consider a system composed by a security camera connected to a gate, using machine learning to unlock the gate and grant access to authorized individuals only. The camera observes the environment* $\mathbf{x}$*, feeding the captured signal* $\mathbf{y}_c$ *to two convolutional neural networks* f *and* g*. The first network implements a binary classifier, which outputs* $f(\mathbf{y}_c) = y_f \in \{authorized, unauthorized\}$*. The network* g *instead predicts the distance* $g(\mathbf{y}_c) = y_g \in \mathbb{R}$ *of the individual from the gate. Finally, the output of the two networks is fed into a simple decision procedure that can unlock the gate,* $d(y_f, y_g) = y_d \in \{locked, unlocked\}$*. While the system is mostly deterministic and can be logically modelled with* $\Delta_{\mathcal{S}}$*, the camera is likely to introduce noise in the captured images. In our framework, this is accounted for and modelled with* $P_{\mathcal{S}}(\mathbf{y}_c \mid \mathbf{x})$*. The verifier provides both a probabilistic model of the environment that the system would observe through the camera,* $P_{\mathcal{R}}(\mathbf{x})$*, as well as one or more logical properties* $\Delta_{\mathcal{R}}$ *that should hold with high probability.*

Following Eq. 2, we notice that the inference problem can be solved by computing the following weighted model integrals:

$$P(\mathcal{R} \mid \mathcal{S}) = \frac{WMI(\Delta_{\mathcal{R}} \wedge \Delta_{\mathcal{S}}, w)}{WMI(\Delta_{\mathcal{S}}, w)} \quad (4)$$

where $\Delta_{\mathcal{R}}$ and $\Delta_{\mathcal{S}}$ are SMT formulas over $\mathbf{e},\mathbf{x},\mathbf{y}$ encoding $\mathcal{R}$ and the deterministic aspect of $\mathcal{S}$ respectively. The weight function encodes the probabilistic aspect of both $\mathcal{S}$ and the environment:

$$w(\mathbf{e},\mathbf{x},\mathbf{y}) = w_{\mathcal{S}}(\mathbf{y} \mid \mathbf{e},\mathbf{x}) \cdot w_{\mathcal{R}}(\mathbf{e},\mathbf{x}) \propto P(\mathbf{e},\mathbf{x},\mathbf{y}).$$

Almost all the properties described below can be expressed in terms of *algebraic* and/or *logical* preconditions and postconditions

($\mathcal{R}_{pre} \rightarrow \mathcal{R}_{post}$). When $\mathcal{R}_{pre}$ is defined on variables in $\mathbf{x}$ only, the verification of a system $\mathcal{S}$ involves computing $P(\mathcal{R}_{post}|\mathcal{R}_{pre}, \mathcal{S})$ (or ratios of these quantities, as shown in Eq. 6 and 7). If instead $\mathcal{R}_{pre}$ also includes variables in $\mathbf{y}$, the quantity $P(\mathcal{R}_{pre} \mid \mathcal{S})$ becomes relevant for the purpose of verifying $\mathcal{S}$, requiring instead the computation of $P(\mathcal{R}_{pre} \rightarrow \mathcal{R}_{post}|\mathcal{S}) = P(\mathcal{R}_{pre}, \mathcal{R}_{post}|\mathcal{S}) + (1 - P(\mathcal{R}_{pre}|\mathcal{S}))$. A WMI-based implementation of the framework can support any of the use case above. For instance, the former case is computed as:

$$P(\mathcal{R}_{post} \mid \mathcal{R}_{pre}, \mathcal{S}) = \frac{WMI(\Delta_{\mathcal{R}_{post}} \wedge \Delta_{\mathcal{R}_{pre}} \wedge \Delta_{\mathcal{S}}, w)}{WMI(\Delta_{\mathcal{R}_{pre}} \wedge \Delta_{\mathcal{S}}, w)} \quad (5)$$

We also observe that properties with no preconditions are special cases of the framework, i.e. $\Delta_{\mathcal{R}_{pre}}$ holds with probability 1. Since SMT($\mathcal{LRA}$) encoding can be naturally conjoined and the class of weight functions supported by the formalism is closed under multiplication, it is trivial to combine arbitrary encodings of $\mathcal{S}$ and $\mathcal{R}$ both from both logical and probabilistic perspectives. Now, we answer the question: "what kind of systems and properties can be encoded in this framework?".

3.1 Probabilistic modelling of $\mathcal{S}$ and $\mathcal{R}$

Prior work on WMI has almost exclusively focused on using polynomials as building blocks for their structured weight functions. The reason is twofold: 1) polynomials are easy to work with, being closed under sum, product and integration over polytopes; 2) they can approximate any density with arbitrary precision. Compared to prior work using piecewise-constant approximations [16, 2], higher-degree polynomials result in either a drastic reduction in the number of pieces for a fixed approximation error or, conversely, in a tighter approximation for a fixed number of pieces (Fig. 3). Notably, the complexity of integration grows only polynomially with respect to the degree [4]. We remark that WMI is not limited to polynomials,

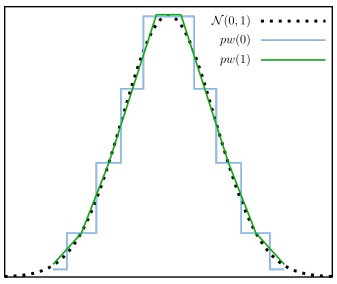

Figure 3. Piecewise approximation of a Gaussian distribution (black dotted line) with degree 0 (light blue) and degree 1 (dark green) polynomials.

any function can be adopted as long as it is integrable, exactly or approximately, in the regions defined via SMT($\mathcal{LRA}$). Gaussians were also employed in the literature, either by restricting to axis-aligned $\mathcal{LRA}$-atoms or by approximating the resulting integrals [37, 20].

The structured representation described in Section 2 has been used to model and reason over a variety of popular probabilistic models, including graphical models like Bayesian and Markov networks [10, 2], probabilistic logic programs with continuous variables [20], and tractable models like density estimation trees (DETs) [45] or sum-product networks (SPNs) [44]. For illustrative purposes, we report the encodings of the latter two models. In both cases, the support $\chi = \bigwedge_{i=1}^{N}(x_i \in [l_i, u_i])$ is typically an-axis-aligned hyperrectagle s.t. $\sum_{\mathbf{A}} \int_{\chi} w(\mathbf{x}, \mathbf{A})\, d\mathbf{x} = 1$, albeit more complex and possibly non-convex supports can be used [40].

DETs The density estimation variant of decision or regression trees shares the same internal structure with binary decision nodes of the form $(x_i \leq k)$ or a proposition A. Different from its predictive variants, each leaf of a DET encodes the probability mass corresponding to the subset of the support induced by the decisions in its path from the root. DETs can be trivially encoded as a tree of nested if-then-elses with constant leaves.

SPNs This class of models combine tractable univariate distributions by means of mixtures (weighted sums) over the same variables or factorizations (products) over disjoint sets of variables, resulting in a tractable but expressive joint probability. The common choices for the univariate distributions, Gaussians and piecewise polynomials [39], as well as their combinations by means of sums and products, are integrable in $\mathcal{LRA}$ and encodable in the formalism.

3.2 Encoding the determinism in $\mathcal{S}$

Besides probabilistic models, other complex functional relationships arising from ML models can be modelled as SMT formulas.

Tree-based predictors As shown above for DETs, the structure of tree-based predictors can be encoded by means of nested if-then-elses with propositional or linear conditions. Leaves are simply encoded with (a conjunction of) atoms $(y = c_i)$, mapping output variables to their respective values. If axis-aligned splits are not expressive enough, arbitrary $\mathcal{LRA}$-conditions enable more complex piecewise decompositions of the joint density, such as those employed in Optimal Classification Trees [12].

Non-linear predictors Support for linear models, which can be trivially encoded in SMT($\mathcal{LRA}$) as $(y = (\mathbf{w}\mathbf{x} + b))$, can be extended to non-linear cases. For instance, we can model neural networks with rectified linear activations [30, 31] by encoding each unit i with inputs $\mathbf{x}_i$, weights $\mathbf{w}_i$ and b_i as:

$$(h_i = (\mathbf{w}_i \mathbf{x}_i + b_i)) \wedge (y_i = ite(h_i > 0,\ h_i,\ 0))$$

The full network is then encoded by conjoining the SMT($\mathcal{LRA}$) representation of each unit. Other non-linear activation functions can be approximated with arbitrary precision. Common operations like convolutions and max/average pooling have SMT($\mathcal{LRA}$) representations. Other non-linear predictors, such as support vector machines with piecewise linear feature maps [27] can be similarly encoded.

Complex models The compositional nature of SMT($\mathcal{LRA}$) can support arbitrary ensembles of K ML models,

$$(\mathbf{y} = a(\mathbf{y}_1, ..., \mathbf{y}_K)) \wedge \bigwedge_{i=1}^{K} \Delta_{f_i}(\mathbf{x})$$

as long as the aggregation function a can be encoded, such as the average. More in general, if every component of an AI system can be modelled with SMT($\mathcal{LRA}$), its behaviour can be verified as a whole. This is in stark contrast with most approaches in literature, which are only able to verify ML components in isolation. As verification of "traditional" software and hardware systems often relies on SMT modelling and solving, using the same paradigm for the verification of modern AI systems is a promising direction.

3.3 *Encoding the properties of $\mathcal{R}$*

So far we demonstrated the flexibility of WMI in modelling and reasoning probabilistically over a wide range of AI systems, but this would be a pointless exercise if we couldn't use it to quantify properties of practical interest. The ML community has identified a number of important properties that learned models should satisfy. For instance, in contexts with high socio-economic stakes, predictions over individuals of a population should be fair. Many definitions of fairness are probabilistic in nature, being based on the notion of *population model* $P(\mathbf{x})$.

Set-based properties For instance, if f is determining a positive vs. negative outcome for an individual [1], like being hired or getting a loan, it is often desirable to quantify the *demographic parity* of a system relative to a protected subpopulation $\mathcal{M}$ [14] by computing the ratio:

$$\frac{P(f(\mathbf{x}) = pos \mid \mathbf{x} \in \mathcal{M})}{P(f(\mathbf{x}) = pos \mid \mathbf{x} \in \overline{\mathcal{M}})} \qquad [\in [1-\delta, 1/(1-\delta)]] \quad (6)$$

In short, the ratio of positive outcomes among members of $\mathcal{M}$ and its complement $\overline{\mathcal{M}}$ should be close. A limitation of this notion is that it does not take into account whether the candidate individuals are qualified for the positive outcome in the first place. If the qualified subpopulation $\mathcal{Q}$ is known to the verifier, the notion of *equality of opportunity* can be quantified instead [25]:

$$\frac{P(f(\mathbf{x}) = pos \mid \mathbf{x} \in \mathcal{M} \cap \mathcal{Q})}{P(f(\mathbf{x}) = pos \mid \mathbf{x} \in \overline{\mathcal{M}} \cap \mathcal{Q})} \qquad [\in [1-\delta, 1/(1-\delta)]] \quad (7)$$

We notice that these preconditions can be trivially encoded when the subsets $\mathcal{M}$ and $\mathcal{Q}$ are characterized by a combination of categorical features, arguably the most common case in fairness scenarios. Additionally, numerical constraints can be encoded with arbitrary piecewise linear sets. For instance, assuming that the output of the model is a binary decision y_{hire}, that $\Delta_{\mathcal{M}} \stackrel{\text{def}}{=} x_{female}$ and that the qualified group can be defined in terms of GPA and years of work experience $\Delta_{\mathcal{Q}} \stackrel{\text{def}}{=} (x_{GPA} > 3.5) \vee (x_{experience} > 20)$, we can compute the ratio in Eq. 7 as [2]:

$$\frac{WMI(y_{hire} \wedge \Gamma, w)}{WMI(\Gamma, w)} \;/\; \frac{WMI(y_{hire} \wedge \Psi, w)}{WMI(\Psi, w)}$$

$$\text{where } \Gamma \stackrel{\text{def}}{=} \Delta_{\mathcal{M}} \wedge \Delta_{\mathcal{Q}} \wedge \Delta_{\mathcal{S}}, \qquad \Psi \stackrel{\text{def}}{=} \neg\Delta_{\mathcal{M}} \wedge \Delta_{\mathcal{Q}} \wedge \Delta_{\mathcal{S}}.$$

Distance-based properties The above notions, which are defined on subpopulations or groups, are commonly referred to as group fairness properties. An alternative notion is *individual fairness* [22], stating that similar individuals, according to a suitable metric d and threshold ϵ, should receive similar treatment/outcomes:

$$P(f(\mathbf{x}) = f(\mathbf{x}') \mid d(\mathbf{x}, \mathbf{x}') < \epsilon) \qquad [> 1-\delta] \quad (8)$$

Notice that the property above is akin to the notion of *probabilistic robustness* [56, 38]. In contrast with fairness verification, where $P(\mathbf{x})$ is defined globally, the distributional assumptions over the perturbations are typically local. For instance, one might verify the robustness of predictions of an image classifier when a Gaussian noise is added to the input instances.

[1] These concepts apply to regression tasks by considering a suitable notion of distance and a threshold.

Encoding distance-based properties poses additional challenges, requiring to define the system's behaviour inside the local neighborhood of infinitely many points. In turn, this requires multiple instantiations of the same random variable $\mathbf{x}, \mathbf{x}' \sim P(\mathbf{x})$ in the SMT($\mathcal{LRA}$) formula. We observe that a simple solution is "cloning" the weight function $w(\mathbf{x})$ and its support Δ:

$$cl(\Delta) \stackrel{\text{def}}{=} \Delta \wedge \Delta', \qquad cl(w) \stackrel{\text{def}}{=} w \cdot w'$$

where $expr'$ denotes the expression obtained by substituting every variable occurrence v with a fresh copy v' in $expr$. This approach, which guarantees that $\mathbf{x}$ and $\mathbf{x}'$ are independently drawn from the same distribution, is called *self-composition* in program analysis [7]. We can then compute queries involving both, such as $x' \in [x-\epsilon, x+\epsilon]$ (Fig. 4).

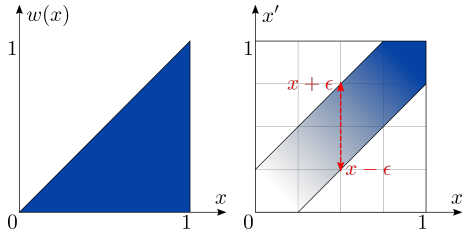

Figure 4. (*Left*) A simple distribution $\Delta = (x \in [0,1])$ and $w(x) = x$. (*Right*) the distribution obtained by self-composition, enabling the computation of queries like $P(x' \in [x-\epsilon, x+\epsilon])$.

In terms of *distances*, both L_1 and L_∞ can be encoded in SMT($\mathcal{LRA}$), by defining absolute values as auxiliary variables $ite((a \leq b), (abs_{a,b} = b - a), (abs_{a,b} = a - b))$:

$$(L_1^{\mathbf{x},\mathbf{x}'} = \sum_{i=1}^{N} abs_{x_i,x_i'}), \qquad (L_\infty^{\mathbf{x},\mathbf{x}'} = max_{i=1}^{N}\, abs_{x_i,x_i'})$$

where $max(.)$ is encoded with nested if-then-elses:

$$max(\{v_1, v_2\}) \stackrel{\text{def}}{=} ite(v_1 < v_2; v_2;\ v_1)$$

$$max(\{v_1, v_2\} \cup V) \stackrel{\text{def}}{=} ite(v_1 < v_2; max(\{v_2\} \cup V); max(\{v_1\} \cup V))$$

Following Eq. 8 and assuming without loss of generality a binary classifier $y = f(\mathbf{x})$, we can quantify individual fairness as:

$$\frac{WMI((y \leftrightarrow y') \wedge (L_\infty^{\mathbf{x},\mathbf{x}'} < \epsilon) \wedge cl(\Delta_{\mathcal{S}}), cl(w))}{WMI((L_\infty^{\mathbf{x},\mathbf{x}'} < \epsilon) \wedge cl(\Delta_{\mathcal{S}}), cl(w))}$$

Probabilistic robustness can be similarly encoded, with the difference that the distribution of noisy inputs $\mathbf{x}'$ is conditioned on $\mathbf{x}$ and explicitly provided as part of $\mathcal{R}$: $w_{\mathcal{R}}(\mathbf{x}, \mathbf{x}') = w_{\mathcal{R}}(\mathbf{x}) \cdot w_{\mathcal{R}}(\mathbf{x}' \mid \mathbf{x})$.

Other algebraic properties In general, with arbitrary combinations of linear inequalities and logical constraints, we can encode many useful algebraic properties. Beyond fairness and robustness, there has been considerable work in enforcing and verifying *monotonic behaviour* of learned predictors [49, 21, 55, 35, 50]. In a probabilistic setting, this can be quantified as $P(f(\mathbf{x}) < f(\mathbf{x}') \,|\, \mathbf{x} < \mathbf{x}')$. Again, this can be computed in WMI by leveraging self-composition:

$$\frac{WMI((y < y') \wedge (\mathbf{x} < \mathbf{x}') \wedge cl(\Delta_{\mathcal{S}}), cl(w))}{WMI((\mathbf{x} < \mathbf{x}') \wedge cl(\Delta_{\mathcal{S}}), cl(w))} \quad (9)$$

Checking the *equivalence* of two predictors f and g finds applications in verifying that a compressed model that has to be deployed in a resource-constrained setting behaves consistently with respect to the original model [43]:

$$P(f(\mathbf{x}) = g(\mathbf{x})) = \frac{WMI((y_f = y_g) \wedge \Delta_{\mathcal{S}}, w)}{WMI(\Delta_{\mathcal{S}}, w)} \tag{10}$$

Monotonicity and equivalence are not the only algebraic properties of interest for the ML community. In their influential paper, Katz et al. ([30]) verify a NN-based collision detection system for unmanned aircrafts against properties such as: "If the intruder is directly ahead and is moving towards the ownship, the system won't issue a clear-of-conflict advisory", encoded as combinations of inequalities over linear and angular quantities. Since all these cyber-physical safety constraints are effectively encoded in SMT($\mathcal{LRA}$), they can be naturally encoded in our framework.

Semantic properties Until now, the properties that we presented are all defined on the *concrete input space* of the system under verification. Yet, being able to define and verify properties using *semantic* features is deemed crucial step for advancing the trustworthiness of our AI sytems [48]. In our conceptual framework, joint logical and probabilistic reasoning over $\mathcal{S}$ and $\mathcal{R}$ is achieved via a unified computational tool, using the same representation language for both. Thus, the classes of ML model that can be encoded in our framework can seamlessly be used when defining both $\mathcal{S}$ and $\mathcal{R}$. For instance, since convolutional NNs with ReLU can be encoded, then $\mathcal{R}$ can also be defined in terms of these models, effectively implementing *neuro-symbolic verification*. Borrowing an example from the seminal work of Xie et al. ([57]), this would enable the verification of properties like "If a stop sign is in front of the camera, a deceleration command is issued" on systems operating at pixel level. The precondition is logical predicate implemented via a binary classifier $\mathcal{R}_{pre} = stop(\mathbf{x})$, mapping images to $true$ if and only if they contain a stop sign. Similarly to $P_{\mathcal{R}}$, the neural predicate $stop(\mathbf{x})$ is part of $\mathcal{R}$ and thus it is assumed to be a certified model with appropriate predictive performance. With our framework, we can quantify $P(f(\mathbf{x}) = decelerate \mid stop(\mathbf{x}))$.

Example 4 *We consider the system described in Ex. 3. Given the high dimensionality of* $\mathbf{x}$*, it would be impossible to define* $P_{\mathcal{R}}(\mathbf{x})$ *manually. Luckily, the verifier provided a validated prior on* $\mathbf{x}$ *(e.g. an SPN) as part of* $\mathcal{R}$*, whose unnormalized density is denoted with* $w_{\mathcal{R}}$*. The noise introduced by the camera can be modelled with a piecewise polynomial approximation of a Gaussian function* $w_{\mathcal{S}}(\mathbf{n}; \sigma) \simeq \mathcal{N}(\mathbf{0}, \sigma\mathbb{I})$ *that conservatively over-estimates the noise level added to each pixel independently:* $\Delta_c \stackrel{def}{=} (\mathbf{y}_c = \mathbf{x} + \mathbf{n})$*. We denote with* Δ_f *and* Δ_g *the SMT($\mathcal{LRA}$) encoding of the CNNs* f *and* g*. Without loss of generality, we assume that* $(y_f \geq 0)$ *iff an individual is* authorized*. The decision procedure* d *simply unlocks the gate if the camera captures an authorized individual that is at most 4 meters away* $\Delta_d \stackrel{def}{=} (y_d \leftrightarrow ((y_f \geq 0) \wedge (y_g \leq 4)))$*. In reality,* d *can be a complex software and or hardware system driven by the predictions. The overall encoding is:*

$$\begin{aligned} \Delta_{\mathcal{S}}(\mathbf{x}, \mathbf{y}) &= \Delta_c(\mathbf{x}, \mathbf{n}, \mathbf{y}_c) \wedge \Delta_d(y_f, y_g, y_d) \\ &\wedge \Delta_f(\mathbf{y}_c, y_f) \wedge \Delta_g(\mathbf{y}_c, y_g) \\ w(\mathbf{x}, \mathbf{n}) &= w_{\mathcal{R}}(\mathbf{x}) \cdot w_{\mathcal{S}}(\mathbf{n}; \sigma) \end{aligned}$$

We can compute how robust are the decisions in noisy (y_d) *vs. noiseless* (y'_d) *settings. To do so, we need self-composition to clone the relevant part of* $\mathcal{S}$*, computing how likely the decision would not change:*

$$WMI((y_d \leftrightarrow y'_d) \wedge \Gamma, w) \quad / \quad WMI(\Gamma, w)$$
$$\textit{where } \Gamma \stackrel{def}{=} (\mathbf{y}'_c = \mathbf{x}) \wedge \Delta_c \wedge cl(\Delta_f \wedge \Delta_g \wedge \Delta_d)$$

Additionally, if we have access to a binary neural predicate $ind(\mathbf{x})$ *that returns* $true$ *iff an individual is present in a frame* $\mathbf{x}$*, we can ensure that the gate is never unlocked when nobody is there:*

$$WMI((y_d \rightarrow ind(\mathbf{x})) \wedge \Delta_{\mathcal{S}}, w) \quad / \quad WMI(\Delta_{\mathcal{S}}, w)$$

4 Conclusion and future work

Our WMI-based perspective on the probabilistic verification of AI systems has many benefits. WMI enables flexible probabilistic reasoning over a broad range of probabilistic models, including distributions over mixed logical/numerical domains (**D1**). These probabilistic models can be combined with multiple machine learning models, using a representation language that is commonly used in PFV. This aspect shows promise in addressing (**D2**). These aspects, paired with the ability to verify many different properties (**D3**), offer unprecedented flexibility in a single framework. Since the focus of this paper is providing a unifying perspective on the problem rather than pushing the scalability of the existing approaches, no empirical evaluation was included. Nonetheless, all the models and properties discussed in the paper could be in principle verified using a single off-the-shelf WMI solver [33]. We conclude by suggesting a number of promising research directions.

Scalability While WMI offers suitable computational means to implement our unifying framework, scaling this approach to large real-world settings requires further work. We have witnessed remarkable progress in solving WMI since its inception, with a multitude of papers focusing on algorithmic and theoretical aspects rather than practical use cases. On the one hand, we hope that this concrete application will become a driving force in the further development of WMI solvers. On the other hand, identifying common factors that hinder the verification might lead to the development of novel models with a favourable trade-off between empirical accuracy and verifiability.

Sequential systems In this work, we focused on non-sequential systems. Yet, most ML models are part of larger software and/or hardware systems with memory. Extending this theoretical framework to sequential systems is thus a fundamental research direction. In terms of implementation, the main advantage of integrating a single interface with respect to implementing multiple system- or property-specific approaches and algorithms, is a less cumbersome and error-prone development process. This makes our framework an appealing candidate to be integrated into existing PFV tools.

Non-linear extensions While WMI is in principle not restricted to SMT($\mathcal{LRA}$) formulas, most works have focused on this setting due to the complexity of reasoning over non-linear constraints while providing bounds on the approximation. Further extensions of WMI to other theories like non-linear algebra, or the support of other families of weight functions, could further push the boundaries of WMI-based verification.

Acknowledgments

We acknowledge the support of the PNRR project FAIR - Future AI Research (PE00000013), under the NRRP MUR program funded by the NextGenerationEU. This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215. The work was also partially supported by the project "AI@TN" funded by the Autonomous Province of Trento.

References

[1] Ralph Abboud, İsmail İlkan Ceylan, and Radoslav Dimitrov, 'Approximate weighted model integration on dnf structures', *Artificial Intelligence*, 103753, (2022).

[2] Aws Albarghouthi, Loris D'Antoni, Samuel Drews, and Aditya V Nori, 'Fairsquare: probabilistic verification of program fairness', *Proceedings of the ACM on Programming Languages*, **1**(OOPSLA), 1–30, (2017).

[3] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané, 'Concrete problems in ai safety', *arXiv preprint arXiv:1606.06565*, (2016).

[4] 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).

[5] Teodora Baluta, Shiqi Shen, Shweta Shinde, Kuldeep S Meel, and Prateek Saxena, 'Quantitative verification of neural networks and its security applications', in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1249–1264, (2019).

[6] Imre Bárány and Zoltán Füredi, 'Computing the volume is difficult', *Discrete & Computational Geometry*, **2**(4), 319–326, (1987).

[7] Gilles Barthe, Pedro R D'argenio, and Tamara Rezk, 'Secure information flow by self-composition', *Mathematical Structures in Computer Science*, **21**(6), 1207–1252, (2011).

[8] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi, 'Measuring neural net robustness with constraints', *Advances in neural information processing systems*, **29**, (2016).

[9] Osbert Bastani, Xin Zhang, and Armando Solar-Lezama, 'Probabilistic verification of fairness properties via concentration', *Proceedings of the ACM on Programming Languages*, **3**(OOPSLA), 1–27, (2019).

[10] Vaishak Belle, Andrea Passerini, and Guy Van den Broeck, 'Probabilistic inference in hybrid domains by weighted model integration', in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, (2015).

[11] Vaishak Belle, Guy Van den Broeck, and Andrea Passerini, 'Hashing-based approximate probabilistic inference in hybrid domains', in *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 141–150. AUAI PRESS, (2015).

[12] Dimitris Bertsimas and Jack Dunn, 'Optimal classification trees', *Machine Learning*, **106**, 1039–1082, (2017).

[13] Rudy R Bunel, Ilker Turkaslan, Philip Torr, Pushmeet Kohli, and Pawan K Mudigonda, 'A unified view of piecewise linear neural network verification', *Advances in Neural Information Processing Systems*, **31**, (2018).

[14] Toon Calders, Faisal Kamiran, and Mykola Pechenizkiy, 'Building classifiers with independency constraints', in *2009 IEEE international conference on data mining workshops*, pp. 13–18. IEEE, (2009).

[15] Hongge Chen, Huan Zhang, Si Si, Yang Li, Duane Boning, and Cho-Jui Hsieh, 'Robustness verification of tree-based models', *Advances in Neural Information Processing Systems*, **32**, (2019).

[16] Dmitry Chistikov, Rayna Dimitrova, and Rupak Majumdar, 'Approximate counting in smt and value estimation for probabilistic programs', in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 320–334. Springer, (2015).

[17] Edmund M Clarke, 'Model checking', in *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pp. 54–56. Springer, (1997).

[18] Luca De Alfaro, *Formal verification of probabilistic systems*, Stanford University, 1998.

[19] Laurens Devos, Wannes Meert, and Jesse Davis, 'Verifying tree ensembles by reasoning about potential instances', in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 450–458. SIAM, (2021).

[20] Pedro Zuidberg Dos Martires, Anton Dries, and Luc De Raedt, 'Exact and approximate weighted model integration with probability density functions using knowledge compilation', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 7825–7833, (2019).

[21] Krishnamurthy Dvijotham, Marta Garnelo, Alhussein Fawzi, and Pushmeet Kohli, 'Verification of deep probabilistic models', *arXiv preprint arXiv:1812.02795*, (2018).

[22] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel, 'Fairness through awareness', in *Proceedings of the 3rd innovations in theoretical computer science conference*, pp. 214–226, (2012).

[23] Gil Einziger, Maayan Goldstein, Yaniv Sa'ar, and Itai Segall, 'Verifying robustness of gradient boosted models', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 2446–2453, (2019).

[24] Timon Gehr, Sasa Misailovic, and Martin Vechev, 'Psi: Exact symbolic inference for probabilistic programs', in *Computer Aided Verification: 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I 28*, pp. 62–83. Springer, (2016).

[25] Moritz Hardt, Eric Price, and Nati Srebro, 'Equality of opportunity in supervised learning', *Advances in neural information processing systems*, **29**, (2016).

[26] Arnd Hartmanns and Holger Hermanns, 'In the quantitative automata zoo', *Science of Computer Programming*, **112**, 3–23, (2015).

[27] Xiaolin Huang, Siamak Mehrkanoon, and Johan AK Suykens, 'Support vector machines with piecewise linear feature mapping', *Neurocomputing*, **117**, 118–127, (2013).

[28] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu, 'Safety verification of deep neural networks', in *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pp. 3–29. Springer, (2017).

[29] Joost-Pieter Katoen, 'The probabilistic model checking landscape', in *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pp. 31–45, (2016).

[30] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer, 'Reluplex: An efficient smt solver for verifying deep neural networks', in *International conference on computer aided verification*, pp. 97–117. Springer, (2017).

[31] Guy Katz, Derek A Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, et al., 'The marabou framework for verification and analysis of deep neural networks', in *International Conference on Computer Aided Verification*, pp. 443–452. Springer, (2019).

[32] Samuel Kolb, Pedro Zuidberg Dos Martires, and Luc De Raedt, 'How to exploit structure while solving weighted model integration problems', in *Uncertainty in Artificial Intelligence*, pp. 744–754. PMLR, (2020).

[33] Samuel Kolb, Paolo Morettin, Pedro Zuidberg Dos Martires, Francesco Sommavilla, Andrea Passerini, Roberto Sebastiani, Luc De Raedt, et al., 'The pywmi framework and toolbox for probabilistic inference using weighted model integration', in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence Demos*, pp. 6530–6532. International Joint Conference on Artificial Intelligence, (2019).

[34] Marta Kwiatkowska, Gethin Norman, and David Parker, 'Stochastic model checking', in *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, pp. 220–270. Springer, (2007).

[35] Xingchao Liu, Xing Han, Na Zhang, and Qiang Liu, 'Certified monotonic neural networks', *Advances in Neural Information Processing Systems*, **33**, 15427–15438, (2020).

[36] Ravi Mangal, Aditya V Nori, and Alessandro Orso, 'Robustness of neural networks: A probabilistic and practical approach', in *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pp. 93–96. IEEE, (2019).

[37] David Merrell, Aws Albarghouthi, and Loris D'Antoni, 'Weighted model integration with orthogonal transformations', in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, (2017).

[38] Matthew Mirman, Alexander Hägele, Pavol Bielik, Timon Gehr, and Martin Vechev, 'Robustness certification with generative models', in

Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, pp. 1141–1154, (2021).

[39] Alejandro Molina, Antonio Vergari, Nicola Di Mauro, Sriraam Natarajan, Floriana Esposito, and Kristian Kersting, 'Mixed sum-product networks: A deep architecture for hybrid domains', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, (2018).

[40] Paolo Morettin, Samuel Kolb, Stefano Teso, and Andrea Passerini, 'Learning weighted model integration distributions', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5224–5231, (2020).

[41] Paolo Morettin, Andrea Passerini, and Roberto Sebastiani, 'Advanced smt techniques for weighted model integration', *Artificial Intelligence*, **275**, 1–27, (2019).

[42] Paolo Morettin, Pedro Zuidberg Dos Martires, Samuel Kolb, and Andrea Passerini, 'Hybrid probabilistic inference with logical and algebraic constraints: a survey', in *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, (2021).

[43] Nina Narodytska, Shiva Kasiviswanathan, Leonid Ryzhyk, Mooly Sagiv, and Toby Walsh, 'Verifying properties of binarized deep neural networks', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, (2018).

[44] Hoifung Poon and Pedro Domingos, 'Sum-product networks: A new deep architecture', in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 689–690. IEEE, (2011).

[45] Parikshit Ram and Alexander G Gray, 'Density estimation trees', in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 627–635, (2011).

[46] Tian Sang, Paul Beame, and Henry A Kautz, 'Performing bayesian inference by weighted model counting', in *AAAI*, volume 5, pp. 475–481, (2005).

[47] Scott Sanner and Ehsan Abbasnejad, 'Symbolic variable elimination for discrete and continuous graphical models', in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, (2012).

[48] Sanjit A Seshia, Dorsa Sadigh, and S Shankar Sastry, 'Towards verified artificial intelligence', *arXiv preprint arXiv:1606.08514*, (2016).

[49] Joseph Sill, 'Monotonic networks', *Advances in neural information processing systems*, **10**, (1997).

[50] Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein, and Guy Van den Broeck, 'Counterexample-guided learning of monotonic neural networks', *Advances in Neural Information Processing Systems*, **33**, 11936–11948, (2020).

[51] Giuseppe Spallitta, Gabriele Masina, Paolo Morettin, Andrea Passerini, and Roberto Sebastiani, 'Smt-based weighted model integration with structure awareness', in *The 38th Conference on Uncertainty in Artificial Intelligence*, (2022).

[52] Roberto Tempo, Er-Wei Bai, and Fabrizio Dabbene, 'Probabilistic robustness analysis: Explicit bounds for the minimum number of samples', in *Proceedings of 35th IEEE Conference on Decision and Control*, volume 3, pp. 3424–3428. IEEE, (1996).

[53] Vincent Tjeng, Kai Xiao, and Russ Tedrake, 'Evaluating robustness of neural networks with mixed integer programming', *arXiv preprint arXiv:1711.07356*, (2017).

[54] Moshe Y Vardi, 'Automatic verification of probabilistic concurrent finite state programs', in *26th Annual Symposium on Foundations of Computer Science (SFCS 1985)*, pp. 327–338. IEEE, (1985).

[55] Antoine Wehenkel and Gilles Louppe, 'Unconstrained monotonic neural networks', *Advances in neural information processing systems*, **32**, (2019).

[56] Lily Weng, Pin-Yu Chen, Lam Nguyen, Mark Squillante, Akhilan Boopathy, Ivan Oseledets, and Luca Daniel, 'Proven: Verifying robustness of neural networks with a probabilistic approach', in *International Conference on Machine Learning*, pp. 6727–6736. PMLR, (2019).

[57] Xuan Xie, Kristian Kersting, and Daniel Neider, 'Neuro-symbolic verification of deep neural networks', (2022).

[58] Zhe Zeng, Paolo Morettin, Fanqi Yan, Antonio Vergari, and Guy Van den Broeck, 'Probabilistic inference with algebraic constraints: Theoretical limits and practical approximations', *Advances in Neural Information Processing Systems*, **33**, 11564–11575, (2020).