

A BIRD’S EYE VIEW ON INFORMED CLASSIFICATION

Anonymous authors

Paper under double-blind review

ABSTRACT

Neurosymbolic AI is a growing field of research aiming to combine neural network learning capabilities with the reasoning abilities of symbolic systems. In this paper, we tackle informed classification tasks, *i.e.*, multi-label classification tasks informed by prior knowledge that specifies which combinations of labels are semantically valid. Several neurosymbolic formalisms and techniques have been introduced in the literature, each relying on a particular language to represent prior knowledge. We take a bird’s eye view on informed classification and introduce a unified formalism that encapsulates all knowledge representation languages. Then, we build upon this formalism to identify several concepts in probabilistic reasoning that are at the core of many techniques across representation languages. We also define a new technique called semantic conditioning at inference, which only constrains the system during inference while leaving the training unaffected, an interesting property in the era of off-the-shelves and foundation models. We discuss its theoretical and practical advantages over two other probabilistic neurosymbolic techniques: semantic conditioning and semantic regularization. We then evaluate experimentally and compare the benefits of all three techniques on several large-scale datasets. Our results show that, despite only working at inference, our technique can efficiently leverage prior knowledge to build more accurate neural-based systems.

1 INTRODUCTION

Neurosymbolic AI is a growing field of research aiming to combine neural network learning capabilities with the reasoning abilities of symbolic systems. This hybridization can take many shapes depending on how the neural and symbolic components interact (Kautz, 2022; Wang et al., 2023). An important sub-field of neurosymbolic AI is Informed Machine Learning (von Rueden et al., 2023), which studies how to leverage prior knowledge to improve neural-based systems. There again, proposed techniques in the literature can be of very different nature depending on the type of task (*e.g.* regression, classification, detection, generation, etc.), the formalism used to represent the prior knowledge (*e.g.* mathematical equations, knowledge graphs, logical theories, etc.), the stage at which knowledge is embedded (*e.g.* data processing, neural architecture design, learning procedure, inference procedure, etc.) and benefits expected from the hybridization (*e.g.* explainability, performance, frugality, etc.).

In particular, informed classification studies multi-label classification tasks where prior knowledge specifies which combinations of labels are semantically valid. In our work, the architecture of the neural model (*e.g.* fully connected, convolutional, transformer-based, etc.) mainly depends on the modality of the input space (*e.g.* images, texts, etc.). Therefore, we consider model-agnostic neurosymbolic techniques that integrate prior knowledge during learning, inference or both, but leave the design of the architecture outside the reach of the technique. To lighten our formalism, we restrict ourselves to the supervised setting throughout the paper, even though some of the techniques we mention can be used in a semi-supervised setting. Several neurosymbolic formalisms and techniques have been introduced in the literature, each using a particular language to represent prior knowledge: HEX-graphs in Deng et al. (2014), propositional formulas in Xu et al. (2018), tractable circuits in Ahmed et al. (2022a), linear programs in Niepert et al. (2021), Prolog in Manhaeve et al. (2021), ASP in Yang et al., First Order Logic in Badreddine et al. (2022), etc. Because of this diversity, it is difficult to identify the shared concepts underlying most techniques introduced in the literature. It is also challenging to establish meaningful comparisons between techniques. In this

paper, we take a bird’s eye view on informed classification and propose a unified formalism that encapsulates all representation languages for propositional knowledge. We build upon this formalism to study a family of neurosymbolic techniques that leverage probabilistic reasoning to integrate prior knowledge, a trend that has gained significant traction in the recent literature (Xu et al., 2018; Manhaeve et al., 2021; Ahmed et al., 2022a;b). We re-frame two existing neurosymbolic techniques: one that only impacts training (semantic regularization) and one that impacts both training and inference (semantic conditioning). We also define a new technique that only impacts the inference stage: **semantic conditioning at inference**. This is a particularly useful property in the era of **off-the-shelves** and **foundation** models (Bommasani et al., 2021), which are pre-trained on massive amounts of general data to then be applied in a multitude of heterogeneous downstream tasks.

Contributions After preliminary notions (Section 2), we propose a unified formalism for **supervised multi-label classification informed by prior knowledge** (Section 3). Then, we introduce **semantic conditioning at inference** (Section 4). To the best of our knowledge, we are the first to define a neurosymbolic technique based on probabilistic reasoning which only impacts inference. We also analyze key properties and the computational complexity of neurosymbolic techniques based on probabilistic reasoning (Section 4.2). Finally, we evaluate the three techniques on several datasets, including large scale datasets whose sizes are rarely encountered in the neurosymbolic literature (Section 5).

2 PRELIMINARIES

In this section, we give the preliminary definitions needed for our unified formalism. We first give a general definition of representation languages for propositional knowledge, which we will use throughout the paper to express prior knowledge on multi-label classification tasks. Then, we define several probabilistic reasoning problems that will be the foundation of the neurosymbolic techniques we will study in the paper.

2.1 KNOWLEDGE REPRESENTATION

In its more abstract form, **knowledge** about a **world** tells us in what **states** this world can be observed. In this paper, we only consider propositional knowledge, where the states correspond to subsets of a discrete set of variables \mathbf{Y} and knowledge tells us what combinations of variables can be observed in the world. The set of *possible* states is $\mathbb{B}^{\mathbf{Y}}$, where $\mathbb{B} := \{0, 1\}$. A state $\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}$ can be seen as a subset of \mathbf{Y} as well as an application that maps each variable to \mathbb{B} (*i.e.*, for a variable $Y_i \in \mathbf{Y}$, $\mathbf{y}_i = 1$ is equivalent to $Y_i \in \mathbf{y}$). Knowledge defines a set of states that are considered *valid*. An *abstract* representation of this knowledge is a **boolean function** $f \in \mathbb{B}^{\mathbb{B}^{\mathbf{Y}}}$, which can be viewed either as a function that maps states in $\mathbb{B}^{\mathbf{Y}}$ to boolean values or as a subset of $\mathbb{B}^{\mathbf{Y}}$. However, in order to exploit this knowledge (*e.g.* reason, query, communicate, etc.), we need a *concrete* language to represent it.

Definition 1 (Propositional language). A **propositional language** is a couple $\mathcal{F} := (\mathcal{T}, \mathcal{J})$ such that for any discrete set of variables \mathbf{Y} :

- $\mathcal{T}(\mathbf{Y})$ is the set of admissible **theories** on \mathbf{Y}
- $\mathcal{J}(\mathbf{Y})$ determines which states on \mathbf{Y} satisfy a theory κ :

$$\mathcal{J}(\mathbf{Y}) : \mathcal{T}(\mathbf{Y}) \rightarrow \mathbb{B}^{\mathbb{B}^{\mathbf{Y}}}$$

When the set of variables is clear from context, we simply note $T \in \mathcal{T}$ and $\mathcal{J}(T)$ in place of $T \in \mathcal{T}(\mathbf{Y})$ and $\mathcal{J}(\mathbf{Y})(T)$. We say that a propositional language $\mathcal{F}_2 := (\mathcal{T}_2, \mathcal{J}_2)$ is a **fragment** of a propositional language $\mathcal{F}_1 := (\mathcal{T}_1, \mathcal{J}_1)$, noted $\mathcal{F}_2 \subset \mathcal{F}_1$, iff for any discrete set of variables \mathbf{Y} : $\mathcal{T}_2(\mathbf{Y}) \subset \mathcal{T}_1(\mathbf{Y})$ and $\mathcal{J}_2(\mathbf{Y})$ is the restriction of $\mathcal{J}_1(\mathbf{Y})$ to $\mathcal{T}_2(\mathbf{Y})$. A state $\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}$ **satisfies** a theory $\kappa \in \mathcal{T}$ iff $\mathbf{y} \in \mathcal{J}(\mathbf{Y})(\kappa)$. We also say that κ accepts \mathbf{y} . A theory κ is **satisfiable** if it is satisfied by a state, *i.e.*, if $\mathcal{J}(\mathbf{Y})(\kappa) \neq \emptyset$. Two theories κ_1 and κ_2 are **equivalent** iff $\mathcal{J}(\kappa_1) = \mathcal{J}(\kappa_2)$.

Definition 1 covers many knowledge representation languages found in the literature. We illustrate below with propositional logic and detail several other propositional languages in Appendix A.

Example 1. A theory in propositional logic is called a **propositional formula** and is formed inductively from variables and other formulas by using unary (\neg , which expresses negation) or binary (\vee , \wedge , which express disjunction and conjunction respectively) connectives. We note $\mathcal{T}_{PL}(\mathbf{Y})$ the set of formulas that can be formed in this way. The semantics of propositional logic can be inductively derived from the formula following the standard semantics of negation, conjunction and disjunction, i.e., a state \mathbf{y} satisfies: a variable $Y_i \in \mathbf{Y}$ if $y_i = 1$, a formula $\neg\phi$ if ϕ is not satisfied by \mathbf{y} , a formula $\phi \vee \psi$ if \mathbf{y} satisfies ϕ or ψ and a formula $\phi \wedge \psi$ if \mathbf{y} satisfies ϕ and ψ . For instance, $\kappa = Y_1 \wedge Y_2$ is satisfied by \mathbf{y} iff $y_1 = y_2 = 1$. We refer to Russell & Norvig (2021) for more details on propositional logic.

2.2 PROBABILISTIC REASONING

One challenge of neurosymbolic AI is to bridge the gap between the discrete nature of logic and the continuous nature of neural networks. Probabilistic reasoning can provide the interface between these two realms by allowing us to reason about uncertain facts.

A probability distribution on a set of **boolean variables** \mathbf{Y} is an application $\mathcal{P} : \mathbb{B}^{\mathbf{Y}} \mapsto \mathbb{R}^+$ that maps each state \mathbf{y} to a probability $\mathcal{P}(\mathbf{y})$, such that $\sum_{\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}} \mathcal{P}(\mathbf{y}) = 1$. To define internal operations between distributions, like multiplication, we extend this definition to un-normalized distributions $\mathcal{E} : \mathbb{B}^{\mathbf{Y}} \mapsto \mathbb{R}^+$. The **null distribution** is the application that maps all states to 0. The **partition function** $Z : \mathcal{E} \mapsto \sum_{\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}} \mathcal{E}(\mathbf{y})$ maps each distribution to its sum, and we note $\bar{\mathcal{E}} := \frac{\mathcal{E}}{Z(\mathcal{E})}$ the normalized distribution (when \mathcal{E} is non-null). The **mode** of a distribution \mathcal{E} is its most probable state, ie $\arg \max_{\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}} \mathcal{E}(\mathbf{y})$.

A standard distribution is the **exponential probability distribution**, which is parameterized by a vector of logits $\mathbf{a} \in \mathbb{R}^k$, one for each variable in \mathbf{Y} , and corresponds to the joint distribution of independent Bernoulli variables $\mathcal{B}(p_i)_{1 \leq i \leq k}$ with $p_i = \text{s}(a_i)$. The independent multi-label classification system (see Example 2) is build by following the probabilistic interpretation based on this distribution.

Definition 2. Given a vector $\mathbf{a} \in \mathbb{R}^k$, the **exponential distribution** is:

$$\mathcal{E}(\cdot|\mathbf{a}) : \mathbf{y} \mapsto \prod_{1 \leq i \leq k} e^{a_i \cdot y_i} \quad (1)$$

We will note $\mathcal{P}(\cdot|\mathbf{a}) = \overline{\mathcal{E}(\cdot|\mathbf{a})}$ the corresponding normalized probability distribution.

Typically, when belief about random variables is expressed through a probability distribution and new information is collected in the form of evidence (i.e., a partial assignment of the variables), we are interested in two things: computing the probability of such evidence and updating our beliefs using Bayes' rules by conditioning the distribution on the evidence. Probabilistic reasoning allows us to perform the same operations with logical knowledge in place of evidence. Let's assume a probability distribution \mathcal{P} on variables $\mathbf{Y} := \{Y_j\}_{1 \leq j \leq k}$ and a **satisfiable** theory κ from a propositional language $\mathcal{F} := (\mathcal{T}, \mathcal{J})$. Notice that the boolean function $\mathcal{J}(\kappa)$ is an un-normalized distribution on \mathbf{Y} .

Definition 3. The **probability** of κ under \mathcal{P} is:

$$\mathcal{P}(\kappa) := Z(\mathcal{P} \cdot \mathcal{J}(\kappa)) = \sum_{\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}} \mathcal{P}(\mathbf{y}) \cdot \mathcal{J}(\kappa)(\mathbf{y}) \quad (2)$$

The distribution \mathcal{P} **conditioned on** κ , noted $\mathcal{P}(\cdot|\kappa)$, is:

$$\mathcal{P}(\cdot|\kappa) := \overline{\mathcal{P} \cdot \mathcal{J}(\kappa)} \quad (3)$$

Since $\mathcal{P}(\cdot|\mathbf{a})$ is strictly positive (for all \mathbf{a}), if κ is satisfiable then its probability under $\mathcal{P}(\cdot|\mathbf{a})$ is also strictly positive. We note:

$$\mathcal{P}(\kappa|\mathbf{a}) := Z(\mathcal{P}(\cdot|\mathbf{a}) \cdot \mathcal{J}(\kappa)) \quad \mathcal{P}(\cdot|\mathbf{a}, \kappa) := \frac{\mathcal{P}(\cdot|\mathbf{a}) \cdot \mathcal{J}(\kappa)}{\mathcal{P}(\kappa|\mathbf{a})}$$

Computing $\mathcal{P}(\kappa|\mathbf{a})$ is a **counting** problem called **Probabilistic Query Estimation** (PQE). Computing the mode of $\mathcal{P}(\cdot|\mathbf{a}, \kappa)$ is an **optimization** problem called **Most Probable Explanation** (MPE). Solving these probabilistic reasoning problems is at the core of many neurosymbolic techniques, as shown in Section 4.

3 INFORMED SUPERVISED CLASSIFICATION

In this section, we introduce a formalism for informed supervised classification. We first detail our computational framework for neural multi-label classification systems, which will serve as our basis for neurosymbolic techniques, then we define what we call informed classification tasks.

3.1 NEURAL MULTI-LABEL CLASSIFICATION

In supervised machine learning, the objective is to learn a relationship between an **input domain** \mathcal{X} and an **output domain** \mathcal{Y} from a labeled dataset $D := (x^i, \mathbf{y}^i)_{1 \leq i \leq d} \in (\mathcal{X} \times \mathcal{Y})^d$.

Deep learning systems usually adopts a *functional framework* to tackle supervised learning tasks. A functional relation $f : \mathcal{X} \mapsto \mathcal{Y}$ is assumed and a neural network (*i.e.*, a parametric and differentiable computational graph) M is designed to model this relation based on assumed properties of f . To learn the parameters of the neural network a differentiable cost function L measuring the distance between predictions and labels is chosen, backpropagation computes the gradient of the loss with respect to each parameter and gradient descent is used to minimize the empirical error. Inference is done by processing the inputs through the neural network.

However, when the output domain \mathcal{Y} is (at least partly) **discrete**, a differentiable distance cannot be defined directly on \mathcal{Y} and such a framework cannot be applied strictly. This is especially relevant for classification tasks. Classification tasks are usually **categorical**: the output domain consists of a finite set of variables. In **multi-label classification** tasks however, elements in the output domain \mathcal{Y} are subsets of a finite set of classes \mathbf{Y} . Usually called **labelsets**, we call them **states** (see Section 2.1) and note $\mathcal{Y} = \mathbb{B}^{\mathbf{Y}}$. Interestingly, categorical classification can be seen as a specific instance of multi-label classification informed with prior knowledge (see Example 4).

Hence, we adopt a slightly modified framework, called *pseudo-functional*, where a third module I (besides M and L), called the inference module, has to be defined to bridge the gap between the continuous nature of the neural network (needed for gradient descent) and the discrete nature of the output space. This third module, although essential, is not often explicitly described. An illustration can be found on Figure 1.

Definition 4. A **neural classification system** for multi-label classification is the given of :

- a **parametric differentiable** (*i.e.*, neural) module M , called the **model**, which takes as inputs $x \in \mathbb{R}^d$, parameters $\theta \in \Theta$ and outputs $M(x, \theta) := M_\theta(x) := \mathbf{a} \in \mathbb{R}^k$, called **pre-activation scores** or **logits**.
- a **non-parametric differentiable** module L , called the **loss** module, which takes $\mathbf{a} \in \mathbb{R}^k$ and $\mathbf{y} \in \{0, 1\}^k$ as inputs and outputs a scalar.
- a **non-parametric** module I , called the **inference** module, which takes $\mathbf{a} \in \mathbb{R}^k$ as input and outputs a prediction $\hat{\mathbf{y}} \in \{0, 1\}^k$.

Remark 1. For lighter notations, we note $\mathbf{a} \in \mathbb{R}^k$ as a simpler notation for $\mathbf{a} \in \mathbb{R}^{\mathbf{Y}}$ assuming $\mathbf{Y} := \{Y_j\}_{1 \leq j \leq k}$.

A common approach to design a neural classification system is to build upon a natural probabilistic interpretation. Logits produced by the neural network are seen as parameters of a conditional probability distribution of the output given the input $\mathcal{P}(\cdot | M_\theta(x))$, the loss module computes the cross-entropy of that distribution with a ground truth label, and the inference module computes the most probable output given the learned distribution.

When no prior knowledge is available about the set of classes (uninformed case), a standard hypothesis is to assume independent output variables. We illustrate below how this translates in a specific neural classification system.

Example 2. For **independent multi-label** classification, we apply a sigmoid layer on logits to turn them into probability scores. The loss is the binary cross-entropy (BCE) between probability scores and labels, and a variable is predicted to be true if its probability is above 0.5 (or equivalently its

logit is above 0). This results in the following modules:

$$\begin{aligned} L_{imc}(\mathbf{a}, \mathbf{y}) &:= \text{BCE}(\mathbf{s}(\mathbf{a}), \mathbf{y}) \\ &= - \sum_j y_j \cdot \log(\mathbf{s}(a_j)) + (1 - y_j) \cdot \log(1 - \mathbf{s}(a_j)) \end{aligned} \quad (4)$$

$$l_{imc}(\mathbf{a}) := \mathbb{1}[\mathbf{a} \geq 0] \quad (5)$$

where $\mathbf{s}(a_i) = \frac{e^{a_i}}{1+e^{a_i}}$ is the sigmoid function and $\mathbb{1}[z] := \begin{cases} 1 & \text{if } z \text{ true} \\ 0 & \text{otherwise} \end{cases}$ the indicator function.

3.2 TASK

A task of supervised multi-label classification is **informed** when it comes attached with prior knowledge, expressed as a theory κ in a propositional language $\mathcal{F} := (\mathcal{T}, \mathcal{J})$, specifying which states in the output space are **semantically valid**.

A supervised dataset $D := (x^i, \mathbf{y}^i)_{1 \leq i \leq d} \in (\mathcal{X} \times \mathcal{Y})^d$ is **consistent** with prior knowledge κ if all labels satisfy κ (i.e., $\forall 1 \leq i \leq n, \mathbf{y}^i \models \kappa$). In this paper we will work under the hypothesis that both training and test datasets are consistent. However, some techniques allow for a relaxation of this assumption, enabling the use of inconsistent datasets.

4 TECHNIQUES

When prior knowledge is available about a classification task, it seems only natural to improve our neural classification system by integrating this knowledge into its design. We give below two examples of informed classification tasks and how the loss and inference modules can be adapted to embed prior knowledge.

Example 3. Categorical classification arises when one and only one output variable is true for a given input sample (e.g. mapping an image to a single digit in $\llbracket 0, 9 \rrbracket$ for MNIST). These constraints can easily be enforced by the following propositional formula:

$$\kappa_{\odot_k} := \left(\bigvee_{1 \leq j \leq k} Y_j \right) \wedge \left(\bigwedge_{1 \leq j < l \leq k} (\neg Y_j \vee \neg Y_l) \right) \quad (6)$$

where the first part ensures that at least one variable is true and the second part prevents two variables to be true simultaneously. For categorical classification, the sigmoid layer is replaced by a softmax layer and the variable with the maximum score is predicted, which leads to the following modules:

$$L_{\odot_k}(\mathbf{a}, \mathbf{y}) := \text{CE}(\mathbf{s}(\mathbf{a}), \mathbf{y}) = -\log(\langle \sigma(\mathbf{a}), \odot_k(j) \rangle) \quad (7)$$

$$l_{\odot_k}(\mathbf{a}) := \odot_k(\arg \max(\mathbf{a})) \quad (8)$$

where CE is the cross-entropy, $\sigma(\mathbf{a}) = \left(\frac{e^{a_j}}{\sum_l e^{a_l}} \right)_{1 \leq j \leq k}$ and \odot_k gives the one-hot encoding (starting at 1) of $j \in \llbracket 1, k \rrbracket$, e.g. $\odot_4(2) = (0, 1, 0, 0)$.

Example 4. Hierarchical classification on a set of variables $\{Y_j\}_{1 \leq j \leq k}$ is usually formulated with a directed acyclic graph $G = (Y, E_h)$ where the nodes are the variables and the edges E_h express subsumption between those variables (e.g. a dog is an animal). This formalism can even be enriched with exclusion edges $H = (Y, E_h, E_e)$ (e.g. an input cannot be both a dog and a cat), like in HEX-graphs (Deng et al., 2014). There again, the translation to propositional logic is straightforward:

$$\kappa_H := \left(\bigwedge_{(i,j) \in E_h} Y_i \vee \neg Y_j \right) \wedge \left(\bigwedge_{(i,j) \in E_e} (\neg Y_i \vee \neg Y_j) \right) \quad (9)$$

where the first part ensures that a son node cannot be true if its father node is not and the second part prevents two mutually exclusive nodes to be true simultaneously. Many techniques have been proposed to integrate hierarchical knowledge in a neural classification system. For instance, (Muller & Smith, 2020) introduces a hierarchical loss to penalize more errors on higher classes of the hierarchy, (Giunchiglia & Lukasiewicz, 2020) refines the logits based on the hierarchy while (Deng et al., 2014) replaces the exponential distribution by a Conditional Random Field that integrates the hierarchical knowledge.

Beyond categorical and hierarchical classification, propositional knowledge can be used to define very diverse output spaces: *e.g.* Sudoku solutions (Augustine et al., 2022), simple paths in a graph (Xu et al., 2018; Ahmed et al., 2022a), preference rankings (Xu et al., 2018), matchings in a graph (Pogačić et al., 2019; Ahmed et al., 2022b), etc.

Therefore, the purpose of a neurosymbolic technique is to automatically derive appropriate loss and inference modules from prior knowledge, **generalizing the work made on categorical and hierarchical cases to arbitrary structures**. We formalize this process with the definition below and illustrate it on Figure 1.

Definition 5 (Neurosymbolic technique). A **neurosymbolic technique** for a propositional language $\mathcal{F} := (\mathcal{T}, \mathcal{J})$ is $\mathfrak{T} := (\mathfrak{L}, \mathfrak{I})$ such that for any finite set of variables \mathbf{Y} and theory $\kappa \in \mathcal{T}(\mathbf{Y})$:

$$\mathfrak{L}(\mathbf{Y}, \kappa) := \mathsf{L} : \mathbb{R}^k \times \mathcal{Y} \mapsto \mathbb{R}^+$$

$$\mathfrak{I}(\mathbf{Y}, \kappa) := \mathsf{I} : \mathbb{R}^k \mapsto \mathcal{Y}$$

Remark 2. A neurosymbolic technique *in general* is not an algorithm, but only a mathematical construct. Therefore, it gives no insight *a priori* into how the modules should be implemented.

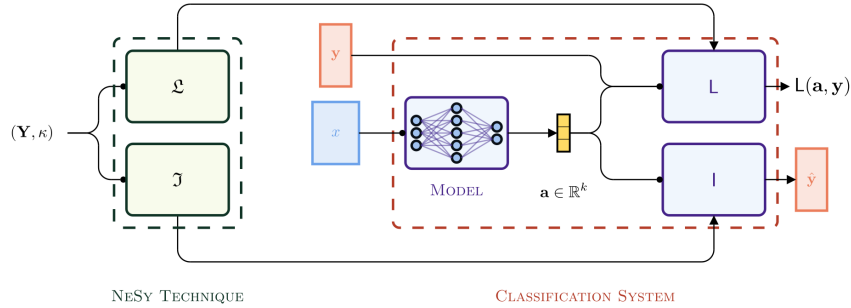


Figure 1: Illustration of a neurosymbolic technique $\mathfrak{T} := (\mathfrak{L}, \mathfrak{I})$: it takes prior knowledge κ as input and outputs the loss L and inference I modules of a neural classification system.

Fuzzy regularization One of the first family of neurosymbolic techniques introduced in the literature was based on fuzzy regularization (Diligenti et al., 2017; Giannini et al., 2023; Badreddine et al., 2022): the fuzzy valuation of prior knowledge κ (expressed in propositional logic) based on output scores $M_\theta(x)$ is added to the standard negative log-likelihood of the labels to steer the model towards *valid* states.

4.1 PROBABILISTIC TECHNIQUES

Besides fuzzy logics, another paradigm that gained traction in the recent years as a basis for designing neurosymbolic techniques is probabilistic reasoning. We define below three neurosymbolic techniques based on probabilistic reasoning, including our new technique called **semantic conditioning at inference**, and relate each technique to the existing neurosymbolic literature.

Semantic regularization Similar to fuzzy regularization, semantic regularization uses the probability of the prior knowledge based on output scores $M_\theta(x)$ (see Section 2.2) as a regularization term. It was introduced for propositional knowledge as the semantic loss in (Xu et al., 2018).

Definition 6. Semantic regularization (with coefficient $\lambda > 0$) for a propositional language $\mathcal{F} := (\mathcal{T}, \mathcal{J})$ is $\mathfrak{T}_r^\lambda := (\mathfrak{L}_r^\lambda, \mathfrak{I}_r^\lambda)$ such that for any finite set of variables \mathbf{Y} and theory $\kappa \in \mathcal{T}(\mathbf{Y})$:

$$\mathfrak{L}_r^\lambda(\mathbf{Y}, \kappa) : (\mathbf{a}, \mathbf{y}) \rightarrow -\log(\mathcal{P}(\mathbf{y}|\mathbf{a})) - \lambda \cdot \log(\mathcal{P}(\kappa|\mathbf{a})) \quad (10)$$

$$\mathfrak{I}_r^\lambda(\mathbf{Y}, \kappa) : \mathbf{a} \rightarrow \arg \max_{\mathbf{y} \in \mathbb{B}^{\mathcal{Y}}} \mathcal{P}(\mathbf{y}|\mathbf{a}) \quad (11)$$

Semantic conditioning Following the probabilistic interpretation mentioned in Section 3.1, a natural way to integrate prior knowledge κ into a neural classification system is to condition the dis-

tribution $\mathcal{P}(\cdot|M(x, \theta))$ on κ . This conditioning affects the loss and inference modules, both underpinned by the conditional distribution. It was first introduced in (Deng et al., 2014) for Hierarchical-Exclusion (HEX) graphs constraints. Semantic probabilistic layers (Ahmed et al., 2022a) can be used to implement semantic conditioning on tractable circuits. NeurASP (Yang et al.) defines semantic conditioning on a predicate extension of ASP programs. An approached method for semantic conditioning on linear programs is proposed in (Niepert et al., 2021).

Definition 7. Semantic conditioning for a propositional language $\mathcal{F} := (\mathcal{T}, \mathcal{J})$ is $\mathfrak{T}_{sc} := (\mathfrak{L}_{sc}, \mathfrak{J}_{sc})$ such that for any finite set of variables \mathbf{Y} and theory $\kappa \in \mathcal{T}(\mathbf{Y})$:

$$\mathfrak{L}_{sc}(\mathbf{Y}, \kappa) : (\mathbf{a}, \mathbf{y}) \rightarrow -\log(\mathcal{P}(\mathbf{y}|\mathbf{a}, \kappa)) \quad (12)$$

$$\mathfrak{J}_{sc}(\mathbf{Y}, \kappa) : \mathbf{a} \rightarrow \arg \max_{\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}} \mathcal{P}(\mathbf{y}|\mathbf{a}, \kappa) \quad (13)$$

Semantic conditioning at inference Finally, we introduce a new neurosymbolic technique, called **semantic conditioning at inference**, which is derived from semantic conditioning but only applies conditioning in the inference module (*i.e.*, infers the most probable state that satisfies prior knowledge) while retaining the standard negative log-likelihood loss.

Definition 8. Semantic conditioning at inference for a propositional language $\mathcal{F} := (\mathcal{T}, \mathcal{J})$ is $\mathfrak{T}_{sci} := (\mathfrak{L}_{sci}, \mathfrak{J}_{sci})$ such that for any finite set of variables \mathbf{Y} and theory $\kappa \in \mathcal{T}(\mathbf{Y})$:

$$\mathfrak{L}_{sci}(\mathbf{Y}, \kappa) : (\mathbf{a}, \mathbf{y}) \rightarrow -\log(\mathcal{P}(\mathbf{y}|\mathbf{a})) \quad (14)$$

$$\mathfrak{J}_{sci}(\mathbf{Y}, \kappa) = \mathfrak{J}_{sc}(\mathbf{Y}, \kappa) \quad (15)$$

4.2 PROPERTIES

As shown above, Definition 5 encapsulates very diverse neurosymbolic techniques. However, beyond this unified view, we can analyze specific properties of neurosymbolic techniques that are critical to their deployment. Formal definitions and proofs can be found in Appendix B.

Syntactic invariance A neurosymbolic technique is **invariant to syntax** when equivalent formulas produce identical loss and inference modules when fed to \mathfrak{L} and \mathfrak{J} . Interestingly, since $\mathfrak{L}(\mathbf{Y}, \kappa)$ and $\mathfrak{J}(\mathbf{Y}, \kappa)$ essentially depend on the boolean function represented by κ and not on the syntax κ , it allows to generalize the technique defined for a particular propositional language to other languages. Because probabilistic reasoning essentially depends on the boolean function represented by the theory κ and not its syntax, techniques that rely on probabilistic reasoning (*e.g.* see Definitions 6, 7 and 8) are *naturally* invariant to syntax. The case of semantic conditioning, which was introduced several times in the neurosymbolic literature using various knowledge representation languages, illustrates the importance of this property and the utility of this unified view of informed classification. On the contrary, because fuzzy regularization is based on the syntax of propositional formulas, it cannot be easily generalized to other representation languages. Besides, this sensitivity to syntax implies that two equivalent propositional formulas (*i.e.*, representing the same prior knowledge) may produce different regularization terms, which could lead to performance disparities that are challenging to elucidate to end users. Finally, even when the mathematical definition of a technique can be easily generalized, this does not mean that its implementation and computational complexity is equivalent regardless of the propositional language.

Consistency A neurosymbolic technique is **consistent** (defined in (Ahmed et al., 2022a)) when the inference module can only produce outputs that satisfy the prior knowledge. By definition, techniques that only impact the loss module (*e.g.* techniques based on regularization terms) cannot verify this property, whereas the inference module of semantic conditioning guarantees consistency.

Besides retaining syntactic invariance and consistency from semantic conditioning, semantic conditioning at inference has other useful properties that make it a suitable choice compared to semantic conditioning and regularization. First, we show in Section 4.3 that it is more tractable computationally. Second, integrating prior knowledge only at inference time offers more flexibility than integration during training. For instance, it can be used if prior knowledge is unavailable at training time (for instance Giunchiglia et al. (2023) provides prior knowledge to an existing task of object detection) or susceptible to evolve. This is a particularly useful property in the era of **off-the-shelves**

and **foundation** models (Bommasani et al., 2021), which are pre-trained on massive amounts of general data to then be applied in a multitude of heterogeneous downstream tasks, since task specific prior knowledge can not be integrated during most of the training process.

4.3 A LOOK ON COMPLEXITY

As mentioned in Section 2.2, all three neurosymbolic techniques defined in Section 4 internally rely on solving MPE and PQE problems. Unfortunately, MPE and PQE are NP-hard and #P-hard respectively for most propositional languages commonly used to represent knowledge (*e.g.* propositional logic, boolean circuits, linear programs, ASP, etc.). This implies that scaling probabilistic neurosymbolic techniques to large classification tasks (*i.e.*, tasks with a large number of variables) on arbitrary prior knowledge requires an exponential amount of computing resources (unless $P = NP$) and is therefore not realistic. Hence, it is critical for any technique to identify (as much as possible) its domain of tractability, which is rarely done in the neurosymbolic literature.

Hopefully, there are fragments of propositional languages for which MPE and PQE are tractable. Boolean circuits in Decomposable Negational Normal Form (DNNF) can solve MPE problems in linear time (in the size of the circuit) and deterministic-DNNF (dDNNF) can additionally solve PQE problems in linear time. An approach that has become predominant in the literature is **knowledge compilation** (Darwiche & Marquis), which consists in translating a theory from an initial propositional language (*e.g.* CNF) into a target propositional language that can solve reasoning problems efficiently (*i.e.*, in a time polynomial in the size of the compiled formula).

Finally, counting problems are known to be much harder in general than optimization problems (Toda, 1991). For instance, MPE can be solved in polynomial time for formulas representing matching constraints (by reduction to finding a maximum weight-sum matching (Edmonds)), while PQE is still #P-hard (Amarilli & Monet). As semantic conditioning at inference only relies on solving MPE for its inference module, compared to semantic conditioning and semantic regularization which both rely on solving PQE to compute their loss module, this implies that semantic conditioning at inference remains tractable for a larger class of tasks than semantic conditioning and semantic regularization.

5 EXPERIMENTS ON LARGE SCALE DATA

In this section, we evaluate empirically the impact of neurosymbolic techniques on four informed classification tasks: a categorical task, two hierarchical tasks and a simple path prediction task.

5.1 A NEW MULTI-SCALE EVALUATION

Most papers in the field evaluate the benefits of their neurosymbolic technique on a single neural network architecture. Although informative, such a methodology paints a very limited picture of the benefits of the technique and leaves many questions unanswered. In particular, it does not allow to estimate how these benefits evolve when resources given to the system (*e.g.* network scale, dataset size, training time, etc.) increase.

To overcome those limitations, we selected for each task a single architectural design that can be scaled to various sizes (*e.g.* DenseNets (Huang et al., 2017)) and compared the performance of the three neurosymbolic techniques against an uninformed baseline (independent multi-label classification) across network scales. We report the **exact accuracy** (Ahmed et al., 2022a) (called coherent accuracy in Deng et al. (2014)), *i.e.*, the share of instances which are well classified on all labels, as our evaluation metric. More details on the methodology and the experimental setup (number of epochs, hyperparameters, etc.) are given in Appendix C.

5.2 DATASETS

We evaluate the three techniques on four different tasks: a categorical task based on MNIST dataset (LeCun et al., 1998), two hierarchical tasks based on Cifar-100 (Krizhevsky, 2009) and ImageNet (Russakovsky et al., 2015) datasets and an acyclic simple path prediction task based on Warcraft Shortest Path (WSP) dataset (Pogančić et al., 2019). For WSP tasks featured in the neurosymbolic

literature (Yang et al.; Niepert et al., 2021; Ahmed et al., 2022a), MPE and PQE are intractable and cannot be scaled to large grids. We modify the WSP dataset to make the graph acyclic, and develop an algorithm to compile acyclic simple path constraints into Ordered Binary Decision Diagrams (a fragment of dDNNF). Therefore, for each type of tasks tackled in our experiments, prior knowledge can be compiled into a polysize dDNNF, allowing to solve MPE and PQE tractably and scale properly with larger of set of variables. See Appendix C.1 for more details.

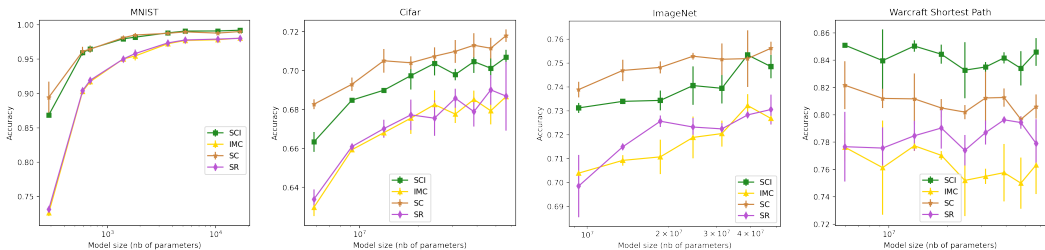


Figure 2: From left to right: each graph plots the exact accuracy on MNIST, Cifar, Imagenet and Warcraft Shortest Path, for all four techniques, against the size of the network. Errorbars represent the standard variation after aggregation on several seeds.

5.3 RESULTS AND ANALYSIS

The results of our experiments are displayed on Figure 2, a graphical representation has been chosen over a tabular one to highlight how accuracy curves evolve as the network scales.

Observation 1. Semantic conditioning and semantic conditioning at inference outperform semantic regularization and independent multi-label classification across tasks and model scales.

Observation 2. Except for the larger networks on Warcraft Shortest Path, semantic regularization brings little benefits in terms of accuracy compared to independent multi-label classification.

Observation 3. On MNIST, Cifar and ImageNet, semantic conditioning at inference retains most of the performance gains (about 75%) of semantic conditioning, despite only integrating knowledge during inference. It even outperforms semantic conditioning on Warcraft Shortest Path.

We expected semantic conditioning to outperform semantic conditioning at inference, since prior knowledge is integrated in the loss module as well as in the inference module, but experiments on Warcraft Shortest Path shows a different picture.

Observation 4. Accuracy gains of semantic conditioning at inference tend to decrease and converge towards a significantly positive value as the accuracy of the neural network increases.

Besides, on MNIST, Cifar and ImageNet, marginal accuracy gains with respect to the network scale are decreasing. In other terms, reaching a 1% accuracy improvement by scaling the network requires much more additional parameters larger networks than for smaller networks. Accuracy gains obtained from the integration of prior knowledge work in a similar fashion: they decrease with the size of the network and converge towards a significantly positive value, meaning that these techniques can improve performances on networks of all sizes. On Warcraft Shortest Path, as the accuracy slightly decreases with network scale, the accuracy gains of semantic conditioning increase with network scale.

The difference of behavior between MNIST, Cifar and ImageNet vs. Warcraft Shortest Path could be due to many factors: the nature of the prior knowledge (categorical and hierarchical vs. simple path), the nature the images (*real life* images vs. synthetic images), etc. More experiences with diverse informed classification tasks are needed to elucidate this point.

Anyhow, our experiments strongly suggests that, in the supervised setting, **the most critical module for knowledge integration is the inference module.**

6 RELATED WORK

In this paper, we restricted our formalism to supervised classification tasks informed with propositional prior knowledge. However, many techniques in the literature work with prior knowledge expressed in a higher order language or solve classification tasks where full supervision is lacking.

Predicate languages Propositional languages use propositional variables to represent atomic facts, which constitute the smallest unit of discourse to represent the world. Predicate languages decompose atomic facts into a more fine grained representation, then leverage this compositional representation through quantification to provide a more expressive language. Predicate languages are often used in the neurosymbolic literature (*e.g.* First Order Logic in (Badreddine et al., 2022), Prolog in (Manhaeve et al., 2021), ASP in (Yang et al.)) to represent compositional knowledge about the input space or impose a structural bias on the neural architecture. However, probabilistic neurosymbolic techniques systematically rely on grounding to perform probabilistic reasoning, which limits the impact of predicate languages on the computational side.

Supervision settings In real world applications, labeling large amounts of data is difficult, expensive and slow, especially for multi-label classification tasks featuring many classes (Deng et al.). Therefore, much work has been done to formalize and exploit cheaper supervision settings where input samples are not fully labeled. In the **semi-supervised** setting (Seeger, 2000; Grandvalet & Bengio, 2004), only a fraction of input samples are fully labeled while the rest is unlabeled. Closely related is the **partial-labels** setting (Durand et al.), where only a subset of the classes are labeled for each input sample. Partial labels can typically be found when prior knowledge represents a functional dependency between a set of latent variables and a set of observed variables, like in the MNIST-Add task (Manhaeve et al., 2021; Badreddine et al., 2022; Maene & De Raedt; van Krieken et al.), which aim is to learn a latent representation of hand-written digits from observing only their sum. Some neurosymbolic techniques have been specifically designed for these supervision settings (Xu et al., 2018; Ahmed et al., 2022b).

7 CONCLUSION

In this paper, we introduce a **formalism for supervised classification informed by prior knowledge**, define a new neurosymbolic technique called **semantic conditioning at inference** which integrates this prior knowledge during inference. To the best of our knowledge, this is the first neurosymbolic technique based on probabilistic reasoning which only impacts inference. We evaluate our technique alongside two existing probabilistic techniques on several large datasets and across neural network scales. We show experimentally that **semantic conditioning at inference can improve the performances of a neural classification system on large datasets and on networks of all sizes**. Besides, we demonstrate that semantic conditioning at inference **preserves key properties** (*i.e.*, syntactic invariance and consistency) and **remains competitive** with semantic conditioning while **only working at inference**, making it **more flexible and tractable**.

Future directions for our work may include, amongst others, reproducing our experiments on more datasets, investigating the semi-supervised and partial-labels settings. Finally, we assume throughout the paper that the knowledge is known *a priori*, which is often not the case in practice. Discovering the *structure* of the task at hand and training the model simultaneously is an important field of research.

REFERENCES

- Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy den Broeck, and Antonio Vergari. Semantic Probabilistic Layers for Neuro-Symbolic Learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 29944–29959. Curran Associates, Inc., 2022a.
- Kareem Ahmed, Eric Wang, Kai-Wei Chang, and Guy Van den Broeck. Neuro-symbolic entropy regularization. In James Cussens and Kun Zhang (eds.), *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pp. 43–53. PMLR, 01–05 Aug 2022b. URL <https://proceedings.mlr.press/v180/ahmed22a.html>.

- 540 Antoine Amarilli and Mikaël Monet. Weighted counting of matchings in unbounded-treewidth graph
541 families. URL <http://arxiv.org/abs/2205.00851>.
542
- 543 Antoine Amarilli, Marcelo Arenas, YooJung Choi, Mikaël Monet, Guy Van den Broeck, and Benjie
544 Wang. A circus of circuits: Connections between decision diagrams, circuits, and automata. *arXiv
545 preprint arXiv:2404.09674*, apr 2024.
- 546 Eriq Augustine, Connor Pryor, Charles Dickens, Jay Pujara, William Yang Wang, and Lise Getoor.
547 Visual sudoku puzzle classification: A suite of collective neuro-symbolic tasks. In *International
548 Workshop on Neural-Symbolic Learning and Reasoning*, 2022.
- 549 Samy Badreddine, Artur d’Avila Garcez, Luciano Serafini, and Michael Spranger. Logic Tensor Net-
550 works. *Artificial Intelligence*, 303:103649, 2022. ISSN 0004-3702. doi: [https://doi.org/10.1016/
551 j.artint.2021.103649](https://doi.org/10.1016/j.artint.2021.103649). URL [https://www.sciencedirect.com/science/article/
552 pii/S0004370221002009](https://www.sciencedirect.com/science/article/pii/S0004370221002009).
- 553 Richard Bellman. On a routing problem. 16(1):87–90. ISSN 0033-569X. URL [https://www.
554 jstor.org/stable/43634538](https://www.jstor.org/stable/43634538). Publisher: Brown University.
- 555 Rishi Bommasani et al. On the opportunities and risks of foundation models. *ArXiv*, 2021. URL
556 <https://crfm.stanford.edu/assets/report.pdf>.
557
- 558 Leo Breiman. Random forests. 45(1):5–32. ISSN 1573-0565. doi: 10.1023/A:1010933404324.
559 URL <https://doi.org/10.1023/A:1010933404324>.
560
- 561 Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge
562 University Press, Cambridge, 2009. ISBN 978-0-521-88438-9. doi:
563 10.1017/CBO9780511811357. URL [https://www.cambridge.org/
564 core/books/modeling-and-reasoning-with-bayesian-networks/
565 8A3769B81540EA93B525C4C2700C9DE6](https://www.cambridge.org/core/books/modeling-and-reasoning-with-bayesian-networks/8A3769B81540EA93B525C4C2700C9DE6).
- 566 Adnan Darwiche and Pierre Marquis. A knowledge compilation map. 17(1):229–264. ISSN 1076-
567 9757.
568
- 569 Jia Deng, Olga Russakovsky, Jonathan Krause, Michael S. Bernstein, Alex Berg, and Li Fei-Fei.
570 Scalable multi-label annotation. In *Proceedings of the SIGCHI Conference on Human Factors
571 in Computing Systems*, CHI ’14, pp. 3099–3102. Association for Computing Machinery. ISBN
572 978-1-4503-2473-1. doi: 10.1145/2556288.2557011. URL [https://doi.org/10.1145/
573 2556288.2557011](https://doi.org/10.1145/2556288.2557011).
- 574 Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut
575 Neven, and Hartwig Adam. Large-Scale Object Classification Using Label Relation Graphs. In
576 *Computer Vision – ECCV 2014*, pp. 48–64. Springer International Publishing, 2014. ISBN 978-
577 3-319-10590-1.
578
- 579 Michelangelo Diligenti, Marco Gori, and Claudio Saccà. Semantic-based regularization for learning
580 and inference. *Artificial Intelligence*, 244:143–165, 3 2017. ISSN 00043702. doi: 10.1016/j.artint.
581 2015.08.011.
- 582 Thibaut Durand, Nazanin Mehrasa, and Greg Mori. Learning a deep ConvNet for multi-label clas-
583 sification with partial labels. URL <http://arxiv.org/abs/1902.09720>.
584
- 585 Jack Edmonds. Maximum matching and a polyhedron with 0,1-vertices. 69B(1):undefined–
586 undefined. ISSN 0022-4340. doi: 10.6028/jres.069b.013. URL [https://www.mendeley.
587 com/catalogue/c0ac3d8d-0ca2-3d92-bd7e-d2570454b3e5/](https://www.mendeley.com/catalogue/c0ac3d8d-0ca2-3d92-bd7e-d2570454b3e5/). Number: 1 and 2.
- 588 Wolfgang Faber. *An Introduction to Answer Set Programming and Some of Its Extensions*, pp. 149–
589 185. Springer International Publishing, Cham, 2020. ISBN 978-3-030-60067-9. doi: 10.1007/
590 978-3-030-60067-9_6. URL https://doi.org/10.1007/978-3-030-60067-9_6.
591
- 592 Yoav Freund and Robert E. Schapire. A desicion-theoretic generalization of on-line learning and
593 an application to boosting. In Paul Vitányi (ed.), *Computational Learning Theory*, pp. 23–37.
Springer. ISBN 978-3-540-49195-8. doi: 10.1007/3-540-59119-2_166.

- 594 Francesco Giannini, Michelangelo Diligenti, Marco Maggini, Marco Gori, and Giuseppe Marra.
595 T-norms driven loss functions for machine learning. *Applied Intelligence*, 53(15):18775–18789,
596 February 2023. ISSN 0924-669X. doi: 10.1007/s10489-022-04383-6. URL <https://doi.org/10.1007/s10489-022-04383-6>.
- 598 Eleonora Giunchiglia and Thomas Lukasiewicz. Coherent hierarchical multi-label classification net-
599 works. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9662–9673. Cur-
600 ran Associates, Inc., 2020. URL [https://proceedings.neurips.cc/paper/2020/](https://proceedings.neurips.cc/paper/2020/hash/6dd4e10e3296fa63738371ec0d5df818-Abstract.html)
601 [hash/6dd4e10e3296fa63738371ec0d5df818-Abstract.html](https://proceedings.neurips.cc/paper/2020/hash/6dd4e10e3296fa63738371ec0d5df818-Abstract.html).
- 603 Eleonora Giunchiglia, Mihaela Cătălina Stoian, Salman Khan, Fabio Cuzzolin, and Thomas
604 Lukasiewicz. ROAD-R: the autonomous driving dataset with logical requirements. *Ma-*
605 *chine Learning*, 112(9):3261–3291, September 2023. ISSN 1573-0565. doi: 10.1007/
606 s10994-023-06322-z. URL <https://doi.org/10.1007/s10994-023-06322-z>.
- 607 Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In
608 *Proceedings of the 17th International Conference on Neural Information Processing Systems*,
609 NIPS’04, pp. 529–536, Cambridge, MA, USA, December 2004. MIT Press.
- 610 Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics,
611 and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman (eds.),
612 *Proceedings of the 7th Python in Science Conference*, pp. 11 – 15, Pasadena, CA USA, 2008.
- 614 Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely Con-
615 nected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern*
616 *Recognition (CVPR)*, pp. 2261–2269, Honolulu, HI, July 2017. IEEE. ISBN 978-1-5386-0457-
617 1. doi: 10.1109/CVPR.2017.243. URL [https://ieeexplore.ieee.org/document/](https://ieeexplore.ieee.org/document/8099726/)
618 [8099726/](https://ieeexplore.ieee.org/document/8099726/).
- 619 Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized
620 neural networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran As-
621 sociates, Inc. URL [https://papers.nips.cc/paper_files/paper/2016/hash/](https://papers.nips.cc/paper_files/paper/2016/hash/d8330f857a17c53d217014ee776bfd50-Abstract.html)
622 [d8330f857a17c53d217014ee776bfd50-Abstract.html](https://papers.nips.cc/paper_files/paper/2016/hash/d8330f857a17c53d217014ee776bfd50-Abstract.html).
- 623 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller, James W.
624 Thatcher, and Jean D. Bohlinger (eds.), *Complexity of Computer Computations: Proceedings*
625 *of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the*
626 *IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the*
627 *Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM*
628 *Research Mathematical Sciences Department*, The IBM Research Symposia Series, pp. 85–103.
629 Springer US. ISBN 978-1-4684-2001-2. doi: 10.1007/978-1-4684-2001-2_9. URL https://doi.org/10.1007/978-1-4684-2001-2_9.
- 631 Henry A. Kautz. The third ai summer: Aai robert s. engelmore memorial lecture. *AI Mag.*, 43:
632 93–104, 2022.
- 633 Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009.
- 635 Martin Krzywinski and Naomi Altman. Classification and regression trees. 14(8):757–758. ISSN
636 1548-7105. doi: 10.1038/nmeth.4370. URL [https://www.nature.com/articles/](https://www.nature.com/articles/nmeth.4370)
637 [nmeth.4370](https://www.nature.com/articles/nmeth.4370). Publisher: Nature Publishing Group.
- 638 Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied
639 to document recognition. *Proceedings of the IEEE*, 86:2278–2323, 1998. ISSN 00189219. doi:
640 10.1109/5.726791.
- 641 Jaron Maene and Luc De Raedt. Soft-unification in deep probabilistic logic. 36.
642 URL [https://papers.nips.cc/paper_files/paper/2023/hash/](https://papers.nips.cc/paper_files/paper/2023/hash/bf215fa7fe70a38c5e967e59c44a99d0-Abstract-Conference.html)
643 [bf215fa7fe70a38c5e967e59c44a99d0-Abstract-Conference.html](https://papers.nips.cc/paper_files/paper/2023/hash/bf215fa7fe70a38c5e967e59c44a99d0-Abstract-Conference.html).
- 645 Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt.
646 Neural probabilistic logic programming in DeepProbLog. *Artificial Intelligence*, 298:103504,
647 September 2021. ISSN 0004-3702. doi: 10.1016/j.artint.2021.103504. URL <https://www.sciencedirect.com/science/article/pii/S0004370221000552>.

- 648 Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, and Marco Gori. Integrating learning
649 and reasoning with deep logic models, 2020.
650
- 651 George A. Miller. Wordnet. *Communications of the ACM*, 38:39–41, 11 1995. ISSN 15577317.
652 doi: 10.1145/219717.219748. URL [https://dl.acm.org/doi/10.1145/219717.](https://dl.acm.org/doi/10.1145/219717.219748)
653 219748.
- 654 Bruce R. Muller and W. Smith. A hierarchical loss for semantic segmentation. In *VISIGRAPP*,
655 2020. URL <https://api.semanticscholar.org/CorpusID:215791996>.
656
- 657 Mathias Niepert, Pasquale Minervini, and Luca Franceschi. Implicit MLE: Backpropa-
658 gating through discrete exponential family distributions. In *Advances in Neural In-*
659 *formation Processing Systems*, volume 34, pp. 14567–14579. Curran Associates, Inc.,
660 2021. URL [https://proceedings.neurips.cc/paper_files/paper/2021/](https://proceedings.neurips.cc/paper_files/paper/2021/hash/7a430339c10c642c4b2251756fd1b484-Abstract.html)
661 [hash/7a430339c10c642c4b2251756fd1b484-Abstract.html](https://proceedings.neurips.cc/paper_files/paper/2021/hash/7a430339c10c642c4b2251756fd1b484-Abstract.html).
- 662 Marin Vlastelica Pogančić, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolínek. Differ-
663 entiation of Blackbox Combinatorial Solvers. September 2019. URL [https://openreview.](https://openreview.net/forum?id=BkevoJSYPB)
664 [net/forum?id=BkevoJSYPB](https://openreview.net/forum?id=BkevoJSYPB).
- 665 J. R. Quinlan. Induction of decision trees. 1(1):81–106. ISSN 1573-0565. doi: 10.1007/
666 BF00116251. URL <https://doi.org/10.1007/BF00116251>.
667
- 668 Ronald L. Rivest. Learning decision lists. 2(3):229–246. ISSN 0885-6125. doi: 10.1023/A:
669 1022607331053. URL <https://doi.org/10.1023/A:1022607331053>.
- 670 Olga Russakovsky et al. Imagenet large scale visual recognition challenge. *International Journal of*
671 *Computer Vision*, 115:211–252, 12 2015. ISSN 15731405. doi: 10.1007/s11263-015-0816-y.
672
- 673 Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach (4th Edition)*. Pearson
674 Higher Ed, 2021.
- 675 Matthias Seeger. Learning with labeled and unlabeled data. 2000.
676
- 677 Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20
678 (5):865–877, 1991. ISSN 0097-5397. doi: 10.1137/0220053. URL [https://epubs.siam.](https://epubs.siam.org/doi/10.1137/0220053)
679 [org/doi/10.1137/0220053](https://epubs.siam.org/doi/10.1137/0220053). Publisher: Society for Industrial and Applied Mathematics.
- 680 Leslie G. Valiant. The complexity of enumeration and reliability problems. 8(3):410–421. ISSN
681 0097-5397. doi: 10.1137/0208032. URL <https://doi.org/10.1137/0208032>.
682
- 683 Emile van Krieken, Thiviyan Thanapalasingam, Jakub Tomczak, Frank van Harmelen, and An-
684 nette Ten Teije. A-NeSI: A scalable approximate method for probabilistic neurosymbolic infer-
685 ence. 36. URL [https://proceedings.neurips.cc/paper_files/paper/2023/](https://proceedings.neurips.cc/paper_files/paper/2023/hash/4d9944ab3330fe6af8efb9260aa9f307-Abstract-Conference.html)
686 [hash/4d9944ab3330fe6af8efb9260aa9f307-Abstract-Conference.html](https://proceedings.neurips.cc/paper_files/paper/2023/hash/4d9944ab3330fe6af8efb9260aa9f307-Abstract-Conference.html).
- 687 Laura von Rueden et al. Informed Machine Learning – A Taxonomy and Survey of Integrating Prior
688 Knowledge into Learning Systems. *IEEE Transactions on Knowledge and Data Engineering*, 35
689 (1):614–633, January 2023. ISSN 1558-2191. doi: 10.1109/TKDE.2021.3079836. Conference
690 Name: IEEE Transactions on Knowledge and Data Engineering.
- 691 Wenguan Wang, Yi Yang, and Fei Wu. Towards data-and knowledge-driven artificial intelligence:
692 A survey on neuro-symbolic computing, 2023. URL [https://arxiv.org/abs/2210.](https://arxiv.org/abs/2210.15889)
693 [15889](https://arxiv.org/abs/2210.15889).
694
- 695 Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van Den Broeck. A semantic loss
696 function for deep learning with symbolic knowledge. In *35th International Conference on Ma-*
697 *chine Learning, ICML 2018*, volume 12, pp. 8752–8760. International Machine Learning Society
698 (IMLS), 2018. ISBN 9781510867963.
- 699 Zhun Yang, Adam Ishay, and Joohyung Lee. NeurASP: Embracing neural networks into answer set
700 programming. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial*
701 *Intelligence*, pp. 1755–1762. International Joint Conferences on Artificial Intelligence Organiza-
tion. ISBN 978-0-9992411-6-5. doi: 10.24963/ijcai.2020/243.

702 A KNOWLEDGE REPRESENTATION

703
704 We detail in this section several other knowledge representation languages. For each language, we
705 first define its syntax then detail its semantics. Other representation languages of boolean func-
706 tions include decision lists (Rivest), decision trees (Krzywinski & Altman; Quinlan), random forests
707 (Breiman), boosted trees (Freund & Schapire) or binarized neural networks (Hubara et al.).
708

709 A.1 CIRCUITS

710
711 Boolean circuits (Darwiche, 2009) $\mathcal{F}_C := (\mathcal{C}, \mathcal{J}_C)$ is a representation language that has gained a
712 lot of traction in recent years because some of its fragments provide tractable algorithms of many
713 reasoning tasks.

714 A **boolean circuit** $C \in \mathcal{C}(\mathbf{Y})$ on variables \mathbf{Y} is a couple $C := (G, \zeta)$ where:

- 715 • $G = (N, W)$ is a directed acyclic graph
- 716 • vertices in N are called **nodes** and edges in W are called **wires**
- 717 • G has a single root r (*i.e.*, a node without parents)
- 718 • $\zeta : N \rightarrow \mathbf{Y} \cup \{\top, \perp, \neg, \wedge, \vee\}$ such that:
 - 719 - $\zeta(n) \in \mathbf{Y} \cup \{\top, \perp\}$ iff n is a leaf node
 - 720 - $\zeta(n) = \neg$ iff n has exactly one child
 - 721 - $\zeta(n) \in \{\wedge, \vee\}$ iff n has at least two children

722
723 The set of children of a node $n \in N$ is noted $ch(n)$. The set of variables of a circuit is noted $var(C)$.
724 Given a node $n \in N$, we note C^n the circuit obtained by keeping all nodes that are descendants of
725 n in G . We sometimes note $var(n)$ for $var(C^n)$.
726
727

728 Let's assume a state $\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}$ and a circuit $C := (G, \zeta) \in \mathcal{C}(\mathbf{Y})$ of root r . To know if \mathbf{y} satisfies the
729 circuit (*i.e.*, $\mathcal{J}_C(C)(\mathbf{y}) = 1$), we evaluate the circuit bottom up, mapping each node n to 0 or 1. First,
730 leaf nodes n are mapped to 1 if $\zeta(n) = \top$, to 0 if $\zeta(n) = \perp$ and to $\mathbf{y}(\zeta(n))$ if $\zeta(n) \in \mathbf{Y}$. Then, for
731 any internal node n , it is valued 1 if $\zeta(n) = \neg$ and its child is valued 1, or if $\zeta(n) = \vee$ and one of its
732 children is valued 1 or if $\zeta(n) = \wedge$ and all its children are valued 1. Otherwise it is valued 0. The
733 state \mathbf{y} satisfies the circuit if the root is valued at 1.

734 A circuit is in **Negational Normal Form** (NNF) if all negation nodes have variables as children,
735 *i.e.*, :

$$736 \forall (u, v) \in W, \zeta(u) = \neg \implies \zeta(v) \in \mathbf{Y}$$

737 (*i.e.*, $\forall (u, v) \in W, \zeta(u) = \neg \implies \zeta(v) \in \mathbf{Y}$).

738 A conjunction node u (*i.e.*, $\zeta(u) = \wedge$) is **decomposable** if the sub-circuits rooted in each of its
739 children do not share variables, *i.e.*, :

$$740 \forall (u, v), (u, w) \in W, var(v) \cap var(w) = \emptyset$$

741
742 A circuit is in **Decomposable Negational Normal Form** (DNNF) if it is NNF and all of its con-
743 junction nodes are decomposable.

744 A disjunction node u (*i.e.*, $\zeta(u) = \vee$) is **deterministic** if the sub-circuits rooted in each of its
745 children do not share satisfying assignments, *i.e.*, :

$$746 \forall (u, v), (u, w) \in W, \mathcal{J}_C(C^v) \cap \mathcal{J}_C(C^w) = \emptyset$$

747
748 A circuit is in **Deterministic Decomposable Negational Normal Form** (dDNNF) if it is DNNF
749 and all of its disjunction nodes are deterministic.

750 Besides, any propositional formula can be translated in linear time into an equivalent boolean circuit
751 by reading the formula in the standard priority order. Therefore, usual fragments of propositional
752 logic (*e.g.* CNF, DNF, etc.) translate into fragments of boolean circuits.

753 Recently, (Amarilli et al., 2024) showed that decision diagrams (*e.g.* Ordered Binary Decision Dia-
754 grams) also correspond to fragments of boolean circuits.

755 A map of fragments of boolean circuits is represented on Figure 3.

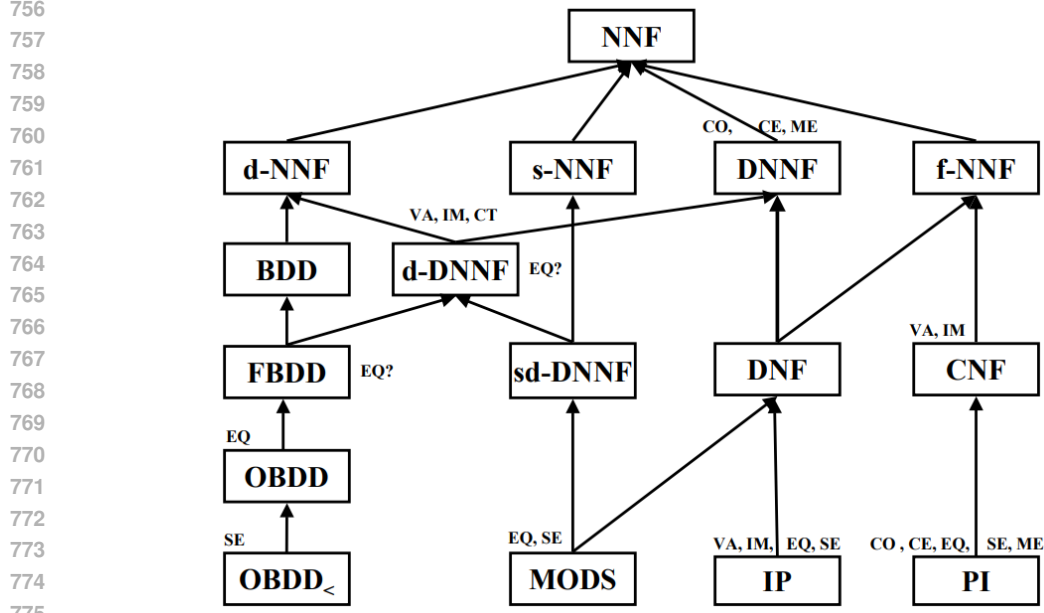


Figure 3: Fragments of boolean circuits in Darwiche (2009): an edge $\mathcal{F}_2 \rightarrow \mathcal{F}_1$ means that \mathcal{F}_2 is a fragment of \mathcal{F}_1 .

A.2 ANSWER SET PROGRAMMING

Answer Set Programming (ASP) (Faber, 2020) $\mathcal{F}_{ASP} := (\mathcal{T}_{ASP}, \mathcal{J}_{ASP})$ is one of the simplest examples of non-monotonic logics, which enable concise representations of complex knowledge at the cost of monotonicity.

A theory $\Pi \in \mathcal{T}_{ASP}(\mathbf{Y})$ in ASP is composed of a set of rules and is called a **program**. A **rule** $r \in \mathcal{R}(\mathbf{Y})$ is formed from the grammar:

$$r ::= a \mid h \leftarrow b \quad h ::= a \mid \perp \quad b ::= l \mid b, l \quad l ::= a \mid \text{not } a$$

where $\{a \in \mathbf{Y}, \leftarrow, \text{not}\}$ are terminal symbols.

h and b are respectively called the **head** and the **body** of the rule.

Example 5. $r := Y_1 \leftarrow Y_2, \text{not } Y_3$ is a rule on variables $\mathbf{Y} = \{Y_1, Y_2, Y_3\}$.

The **reduct** of a program $\Pi \in \mathcal{T}_{ASP}(\mathbf{Y})$ relative to a state $\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}$ is the program $\Pi^{\mathbf{y}}$:

$$\Pi^{\mathbf{y}} = \{Y_{i_0} \leftarrow Y_{i_1}, \dots, Y_{i_l} \mid r \in \Pi, r : Y_{i_0} \leftarrow Y_{i_1}, \dots, Y_{i_l}, \text{not } Y_{i_{l+1}}, \dots, \text{not } Y_{i_m}, \forall j \in \{i_{l+1}, \dots, i_m\}, y_j = 0\}$$

To get $\Pi^{\mathbf{y}}$, we first eliminate all rules in Π such that \mathbf{y} does not satisfy the negative part of the body, then for remaining rules, we delete the negative part of the body and add them to $\Pi^{\mathbf{y}}$.

We say that a state $\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}$ **fires** a rule $r : Y_{i_0} \leftarrow Y_{i_1}, \dots, Y_{i_l}, \text{not } Y_{i_{l+1}}, \dots, \text{not } Y_{i_m}$ iff \mathbf{y} satisfies ϕ_r (in the sense of propositional logic) where:

$$\phi_r := \neg Y_{i_1} \vee \dots \vee \neg Y_{i_l} \vee Y_{i_{l+1}} \vee Y_{i_m} \vee \neg Y_{i_0}$$

Finally, \mathcal{J}_{ASP} follows the **answer set semantics**: a state $\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}$ in an **answer set** for a program Π , noted $\mathcal{J}_{ASP}(\Pi)(\mathbf{y}) = 1$, iff it is the smaller state (in terms of inclusion) to fire all rules of $\Pi^{\mathbf{y}}$.

A.3 LINEAR PROGRAMMING

Linear programming is traditionally associated to constrained optimization problems, but it can be used to define a propositional language $\mathcal{F}_{LP} := (\mathcal{T}_{LP}, \mathcal{J}_{LP})$ naturally suited to express many real-world problems.

A theory $\Pi \in \mathcal{T}_{LP}$, called a **linear program**, is a set of formulas called **linear constraints**. A linear constraint $r \in \mathcal{LC}(\mathbf{Y})$ is of the shape:

$$b_1 \cdot Y_{i_1} + \dots + b_m \cdot Y_{i_m} \leq c$$

with $Y_{i_1}, \dots, Y_{i_m} \in \mathbf{Y}$ and $b_1, \dots, b_m, c \in \mathbb{Z}$.

To lighten notations, a linear constraint is sometimes noted $\langle \mathbf{Z}, \mathbf{b} \rangle = c$ where $\mathbf{Z} := (Y_{i_1}, \dots, Y_{i_m})$ and $\mathbf{b} := (b_1, \dots, b_m)$.

Remark 3. The set of coefficients \mathbb{Z} can be extended to \mathbb{Q} without affecting either the concision of the language nor its expressivity. However, it cannot be extended to \mathbb{R} because arbitrary irrational coefficients would require an infinite length to represent.

A state $\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}$ **satisfies** a linear constraint iff $b_1 \cdot y_{i_1} + \dots + b_m \cdot y_{i_m} \leq c$ in the usual arithmetical sense. A state $\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}$ satisfies a linear program $\Pi \in \mathcal{T}_{LP}$ (i.e., $\mathcal{J}_{LP}(\Pi)(\mathbf{y}) = 1$) iff it satisfies all linear constraints in Π .

Example 6. *Imagine a catalog of products $\mathbf{P} := \{P_1, \dots, P_k\}$ with corresponding prices $\mathbf{p} := \{p_1, \dots, p_k\} \in \mathbb{N}^k$. A basket of products corresponds to a state on \mathbf{P} . An online website might want to suggest a basket of additional products to go with the order of a client. However, it noticed that large or expensive baskets are less likely to be picked. However, they would like to make sure that the suggested basket is not too cheap. Therefore, they defined a maximum size N_M as well as maximum and minimum budgets B_M and B_m for the suggested baskets. The set of baskets that match those constraints correspond to a boolean function on \mathbf{P} .*

This boolean function can be represented by the following linear program:

$$\langle \mathbf{P}, \mathbf{1} \rangle \leq N_M$$

$$\langle \mathbf{P}, \mathbf{p} \rangle \leq B_M$$

$$\langle \mathbf{P}, \mathbf{p} \rangle \geq B_m$$

A.4 GRAPH-BASED LANGUAGES

Although they are not usually thought of as propositional languages, graphs can allow to express knowledge about variables by relating them to elements of a graph. As opposed to most propositional languages, graph-based languages are not universal (i.e., they cannot represent any boolean function), but specialized for a specific type of knowledge. They are often used to represent fragments of universal languages in a concise and more intuitive way. Besides, unlike most propositional languages where the semantic is strongly tied to the syntax (such that the semantics is often implicitly assumed from the syntax), graph-based languages share very similar syntaxes but vary greatly in their semantics.

A language is **edge-based** (resp. **vertice-based**) when a theory maps variables in \mathbf{Y} to edges (resp. vertices) of a graph $G = (V, E)$: a theory is a couple $T := (G, \zeta)$ where $G = (V, E)$ is a graph and $\zeta : E \rightarrow \mathbf{Y}$ (resp. $\zeta : V \rightarrow \mathbf{Y}$) is **bijective**.

Example 7. *The simple path language (resp. matching language) is an edge-based language where a state $\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}$ satisfies T iff the set of selected edges forms a simple path (resp. perfect matching) in G .*

B PROPERTIES

We give formal definitions and proofs for syntactic invariance and consistency of neurosymbolic techniques mentioned in the paper. Besides, we demonstrate several other properties of interest of probabilistic neurosymbolic techniques. We underline the generality of semantic conditioning by showing that traditional loss and inference modules introduced for independent, categorical and hierarchical classification are specific cases of semantic conditioning on their respective semantics. We demonstrate that conditioning at inference is *superior* to independent inference in a certain sense that we define.

864 B.1 SYNTACTIC INVARIANCE

865
866 **Definition 9** (Syntactic invariance). A model agnostic neurosymbolic technique $\mathfrak{T} := (\mathfrak{L}, \mathfrak{J})$ for a
867 propositional language $\mathcal{F} := (\mathcal{T}, \mathcal{J})$ is **invariant to syntax** iff, for any finite set of variables \mathbf{Y} and
868 theories $\kappa_1, \kappa_2 \in \mathcal{T}(\mathbf{Y})$ such that $\kappa_1 \equiv \kappa_2$:

$$\begin{aligned} 869 \mathfrak{L}(\mathbf{Y}, \kappa_1) &= \mathfrak{L}(\mathbf{Y}, \kappa_2) \\ 870 \mathfrak{J}(\mathbf{Y}, \kappa_1) &= \mathfrak{J}(\mathbf{Y}, \kappa_2) \end{aligned}$$

871
872 **Proposition 1.** *Semantic regularization, semantic conditioning and semantic conditioning at infer-*
873 *ence are all invariant to syntax.*

874
875 *Proof.* Like mentioned in the paper, this stems directly from the fact that probabilistic reasoning
876 essentially depends on the semantic of the formula rather than on its syntax. By definition, for
877 $\kappa_1 \equiv \kappa_2$ we have $\mathbb{1}_{\kappa_1} = \mathbb{1}_{\kappa_2}$. This implies that for any $\mathbf{a} \in \mathbb{R}^k$ we have $\mathcal{P}(\kappa_1|\mathbf{a}) = \mathcal{P}(\kappa_2|\mathbf{a})$ and
878 $\arg \max_{\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}} \mathcal{P}(\cdot|\mathbf{a}, \kappa_1) = \arg \max_{\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}} \mathcal{P}(\cdot|\mathbf{a}, \kappa_2)$, which concludes the proof for all three techniques. \square
879

880 B.2 CONSISTENCY

881
882 **Definition 10** (Consistency). A model agnostic neurosymbolic technique $\mathfrak{T} := (\mathfrak{L}, \mathfrak{J})$ for a propo-
883 sitional language $\mathcal{F} := (\mathcal{T}, \mathcal{J})$ is **consistent** iff, for any finite set of variables \mathbf{Y} and a satisfiable
884 theory $\kappa \in \mathcal{T}(\mathbf{Y})$:

$$885 \forall \mathbf{a} \in \mathbb{R}^k, \mathfrak{J}(\mathbf{Y}, \kappa)(\mathbf{a}) \models \kappa$$

886 **Proposition 2.** *Both semantic conditioning and semantic conditioning at inference are consistent.*

887
888 *Proof 2.* Remind that $\mathfrak{J}_{sc}(\mathbf{Y}, \kappa)(\mathbf{a}) = \arg \max_{\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}} \mathcal{P}(\mathbf{y}|\mathbf{a}, \kappa)$.

889
890 We assumed κ to be satisfiable and we know that for all \mathbf{a} , $\mathcal{P}(\cdot|\mathbf{a})$ is strictly positive. Therefore
891 $\mathcal{P}(\cdot|\mathbf{a}, \kappa)$ is strictly positive and we have:

$$892 \mathbf{y} = \arg \max_{\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}} \mathcal{P}(\mathbf{y}|\mathbf{a}, \kappa) \implies \mathcal{P}(\mathbf{y}|\mathbf{a}, \kappa) > 0 \implies \mathbf{y} \models \kappa$$

893 Hence:

$$894 \forall \mathbf{a}, \mathfrak{J}_{sc}(\mathbf{Y}, \kappa)(\mathbf{a}) \models \kappa$$

895 \square

896 B.3 GENERALITY OF SEMANTIC CONDITIONING

897
898 First, let us demonstrate that standard modules for independent and categorical classification are
899 particular cases of semantic conditioning on their respective background knowledge:

900 **Proposition 3.**

$$\begin{aligned} 901 \mathfrak{L}_{sc}(\mathbf{Y}, \top)(\mathbf{a}, \mathbf{y}) &= \mathbb{L}_{imc}(\mathbf{a}, \mathbf{y}) & \mathfrak{J}_{sc}(\mathbf{Y}, \top)(\mathbf{a}) &= \mathbb{I}_{imc}(\mathbf{a}) \\ 902 \mathfrak{L}_{sc}(\mathbf{Y}, \kappa_{\odot_k})(\mathbf{a}, \mathbf{y}) &= \mathbb{L}_{\odot_k}(\mathbf{a}, \mathbf{y}) & \mathfrak{J}_{sc}(\mathbf{Y}, \kappa_{\odot_k})(\mathbf{a}) &= \mathbb{I}_{\odot_k}(\mathbf{a}) \end{aligned} \quad (16)$$

903
904 We start by demonstrating the following lemma:

905 **Lemma 4.** *Let's assume $\mathbf{a} \in \mathbb{R}^k$, then:*

$$906 \mathcal{P}(\mathbf{y}|\mathbf{a}) = \prod_{1 \leq j \leq k} y_j \cdot s(a_j) + (1 - y_j) \cdot (1 - s(a_j))$$

907
908 where $s(\mathbf{a}) = (\frac{e^{a_j}}{1+e^{a_j}})_{1 \leq j \leq k}$ is the sigmoid function.

909 *Proof.* To prove this, let's prove by recurrence on $k \in \mathbb{N}^*$ that:

$$910 \forall \mathbf{a} \in \mathbb{R}^k, Z(\mathcal{E}(\cdot|\mathbf{a})) = \prod_{1 \leq j \leq k} (1 + e^{a_j})$$

918 First, let's assume $k = 1$, we have:

$$919 \quad \forall a \in \mathbb{R}, Z(\mathcal{E}(\cdot|a)) = \mathcal{E}(0|a) + \mathcal{E}(1|a) = e^0 + e^a = 1 + e^a$$

921 Then, let's assume $k > 1$, we have:

$$922 \quad \forall \mathbf{a} \in \mathbb{R}^k, Z(\mathcal{E}(\cdot|\mathbf{a})) = \sum_{\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}} \mathcal{E}(\mathbf{y}|\mathbf{a}) = \sum_{\mathbf{y} \in \mathbb{B}^{\mathbf{Y}}} \prod_{1 \leq i \leq k} e^{a_i \cdot y_i}$$

$$923 \quad = \sum_{\substack{\mathbf{y} \in \mathbb{B}^{\mathbf{Y}} \\ y_k = 0}} \prod_{1 \leq i \leq k-1} e^{a_i \cdot y_i} + \sum_{\substack{\mathbf{y} \in \mathbb{B}^{\mathbf{Y}} \\ y_k = 1}} e^{a_k} \cdot \prod_{1 \leq i \leq k-1} e^{a_i \cdot y_i}$$

$$924 \quad = (1 + e^{a_k}) \cdot \sum_{\mathbf{y} \in \mathbb{B}^{k-1}} \prod_{1 \leq i \leq k-1} e^{a_i \cdot y_i} = (1 + e^{a_k}) \cdot Z(\mathcal{E}(\cdot|\mathbf{a}_{\setminus k}))$$

930 where $\mathbf{a}_{\setminus k} = (a_j)_{1 \leq j \leq k-1}$.

932 By application of the recurrence hypothesis:

$$933 \quad Z(\mathcal{E}(\cdot|\mathbf{a}_{\setminus k})) = \prod_{1 \leq j \leq k-1} (1 + e^{a_j})$$

936 Hence:

$$937 \quad \forall \mathbf{a} \in \mathbb{R}^k, Z(\mathcal{E}(\cdot|\mathbf{a})) = \prod_{1 \leq j \leq k} (1 + e^{a_j})$$

939 This gives us:

$$940 \quad \forall \mathbf{y} \in \mathbb{B}^{\mathbf{Y}}, \forall \mathbf{a} \in \mathbb{R}^k, \mathcal{P}(\mathbf{y}|\mathbf{a}) = \frac{\mathcal{E}(\mathbf{y}|\mathbf{a})}{Z(\mathcal{E}(\cdot|\mathbf{a}))} = \frac{\prod_{1 \leq i \leq k} e^{a_i \cdot y_i}}{\prod_{1 \leq j \leq k} (1 + e^{a_j})} = \prod_{1 \leq j \leq k} \frac{e^{a_j \cdot y_j}}{1 + e^{a_j}}$$

943 Notice that:

$$944 \quad \forall y \in \mathbb{B}, a \in \mathbb{R}, \frac{e^{a \cdot y}}{1 + e^a} = y \cdot s(a) + (1 - y) \cdot (1 - s(a))$$

946 Thus, finally:

$$947 \quad \forall \mathbf{y} \in \mathbb{B}^{\mathbf{Y}}, \forall \mathbf{a} \in \mathbb{R}^k, \mathcal{P}(\mathbf{y}|\mathbf{a}) = \prod_{1 \leq j \leq k} y_j \cdot s(a_j) + (1 - y_j) \cdot (1 - s(a_j))$$

950 \square

952 *Proof 3.1.* First, according to Lemma 4:

$$953 \quad \mathcal{P}(\mathbf{y}|\mathbf{a}) = \prod_{1 \leq j \leq k} y_j \cdot s(a_j) + (1 - y_j) \cdot (1 - s(a_j))$$

956 Besides, we know that $\mathbb{1}_{\top} = 1$ (all states are mapped to 1), which implies that:

$$957 \quad \forall \mathbf{y} \in \mathbb{B}^{\mathbf{Y}}, \forall \mathbf{a} \in \mathbb{R}^k, \mathcal{P}(\mathbf{y}|\mathbf{a}, \top) = \mathcal{P}(\mathbf{y}|\mathbf{a})$$

959 This gives:

$$960 \quad \mathcal{L}_{sc}(\mathbf{Y}, \top)(\mathbf{a}, \mathbf{y}) = -\log(\mathcal{P}(\mathbf{y}|\mathbf{a}, \top)) = -\log(\mathcal{P}(\mathbf{y}|\mathbf{a}))$$

$$961 \quad = -\log\left(\prod_j y_j \cdot s(a_j) + (1 - y_j) \cdot (1 - s(a_j))\right)$$

$$962 \quad = -\sum_j \log(y_j \cdot s(a_j) + (1 - y_j) \cdot (1 - s(a_j)))$$

967 Since \mathbf{y} is a binary vector:

$$968 \quad \mathcal{L}_{sc}(\mathbf{Y}, \top)(\mathbf{a}, \mathbf{y}) = -\sum_j y_j \cdot \log(s(a_j)) + (1 - y_j) \cdot \log(1 - s(a_j))$$

$$969 \quad = \mathcal{L}_{imc}(\mathbf{a}, \mathbf{y})$$

971 \square

972 *Proof 3.2.*

$$\begin{aligned}
973 \quad \mathfrak{J}_{sc}(\mathbf{Y}, \top)(\mathbf{a}) &= \arg \max_{\mathbf{y} \in \mathbb{B}^Y} \mathcal{P}(\mathbf{y}|\mathbf{a}, \top) = \arg \max_{\mathbf{y} \in \mathbb{B}^Y} \mathcal{P}(\mathbf{y}|\mathbf{a}) = \arg \max_{\mathbf{y} \in \mathbb{B}^Y} \mathcal{E}(\mathbf{y}|\mathbf{a}) \\
974 &= \arg \max_{\mathbf{y} \in \mathbb{B}^Y} \prod_{1 \leq i \leq k} e^{a_i \cdot y_i} = \arg \max_{\mathbf{y} \in \mathbb{B}^Y} [\exp(\sum_{1 \leq i \leq k} a_i \cdot y_i)] = \arg \max_{\mathbf{y} \in \mathbb{B}^Y} \sum_{1 \leq i \leq k} a_i \cdot y_i \\
975 &= \mathbf{1}[\mathbf{a} \geq 0] \\
976 &= l_{imc}(\mathbf{a})
\end{aligned}$$

□

982 *Proof 3.3.* The one and only one semantic of κ_{\odot_k} gives us:

$$983 \quad \forall \mathbf{y}, \mathbf{y} \models \kappa_{\odot_k} \implies \exists j, \mathbf{y} = \odot_k(j)$$

984 Hence:

$$\begin{aligned}
985 \quad \mathcal{P}(\kappa_{\odot_k}|\mathbf{a}) &= \sum_{\mathbf{y} \models \kappa_{\odot_k}} \mathcal{P}(\mathbf{y}|\mathbf{a}) = \sum_{1 \leq j \leq k} \mathcal{P}(\odot_k(j)|\mathbf{a}) \\
986 &= \frac{1}{Z(\mathcal{P}(\cdot|\mathbf{a}))} \cdot \sum_{1 \leq j \leq k} \mathcal{E}(\odot_k(j)|\mathbf{a}) = \frac{1}{Z(\mathcal{P}(\cdot|\mathbf{a}))} \cdot \sum_{1 \leq j \leq k} e^{a_j}
\end{aligned}$$

987 This leads to:

$$\begin{aligned}
988 \quad \forall l, \mathcal{P}(\odot_k(l)|\mathbf{a}, \kappa_{\odot_k}) &= \frac{\mathcal{P}(\odot_k(l) \wedge \kappa_{\odot_k}|\mathbf{a})}{\mathcal{P}(\kappa_{\odot_k}|\mathbf{a})} = \frac{\mathcal{P}(\odot_k(l)|\mathbf{a})}{\mathcal{P}(\kappa_{\odot_k}|\mathbf{a})} = \frac{\mathcal{E}(\odot_k(l)|\mathbf{a})}{Z(\mathcal{P}(\cdot|\mathbf{a})) \cdot \mathcal{P}(\kappa_{\odot_k}|\mathbf{a})} \\
989 &= \frac{e^{a_l}}{\sum_{1 \leq j \leq k} e^{a_j}} = \sigma(\mathbf{a})_l = \langle \sigma(\mathbf{a}), \odot_k(l) \rangle
\end{aligned}$$

990 Besides, since we assume consistent labels, we know that there is l such that $\mathbf{y} = \odot_k(l)$, which gives:

$$\begin{aligned}
991 \quad \mathfrak{L}_{sc}(\mathbf{Y}, \kappa_{\odot_k})(\mathbf{a}, \mathbf{y}) &= L_{\kappa_{\odot_k}}(\mathbf{a}, \odot_k(l)) = -\log(\mathcal{P}(\odot_k(l)|\mathbf{a}, \kappa_{\odot_k})) \\
992 &= -\log(\langle \sigma(\mathbf{a}), \odot_k(l) \rangle) = -\log(\langle \sigma(\mathbf{a}), \mathbf{y} \rangle) \\
993 &= L_{\odot_k}(\mathbf{a}, \mathbf{y})
\end{aligned}$$

□

1000 *Proof 3.4.* We know that κ_{\odot_k} is satisfiable and $\mathcal{P}(\mathbf{y}|\mathbf{a})$ is strictly positive. So we have:

$$\begin{aligned}
1001 \quad \mathbf{y} &= \arg \max_{\mathbf{y} \in \mathbb{B}^Y} \mathcal{P}(\mathbf{y}|\mathbf{a}, \kappa_{\odot_k}) \implies \mathbf{y} \models \kappa_{\odot_k} \\
1002 &\implies \exists l, \mathbf{y} = \odot_k(l)
\end{aligned}$$

1003 Therefore, we have:

$$\begin{aligned}
1004 \quad \mathfrak{J}_{sc}(\mathbf{Y}, \kappa_{\odot_k})(\mathbf{a}) &= \arg \max_{1 \leq j \leq k} \mathcal{P}(\odot_k(j)|\mathbf{a}, \kappa_{\odot_k}) = \arg \max_{1 \leq l \leq k} \mathcal{P}(\odot_k(l)|\mathbf{a}) \\
1005 &= \arg \max_{1 \leq l \leq k} \langle \mathbf{a}, \odot_k(l) \rangle = \odot_k(\arg \max_{1 \leq l \leq k} \langle \mathbf{a}, \odot_k(l) \rangle) \\
1006 &= l_{\odot_k}(\mathbf{a})
\end{aligned}$$

□

1022 B.4 SUPERIORITY OF CONDITIONING AT INFERENCE

1023 Besides, when performing inference based on identical model modules and learned parameters, *sci* **guarantees** greater or equal accuracy compared to traditional *imc* inference (*i.e.*, if l_{imc} infers the right labels, then $\mathfrak{J}_{sc}(\mathbf{Y}, \kappa)$ will also infer the right labels):

1026 **Proposition 5.**

$$1027 \quad \forall \mathbf{a} \in \mathbb{R}^k, \mathbf{y} \models \kappa, \mathbf{l}_{imc}(\mathbf{a}) = \mathbf{y} \implies \mathcal{J}_{sci}(\mathbf{Y}, \kappa)(\mathbf{a}) = \mathbf{y}$$

1028
1029 *Proof 5.* Let’s prove this by the absurd and assume that:

$$1030 \quad \mathcal{J}_{sc}(\mathbf{Y}, \kappa)(\mathbf{a}) := \hat{\mathbf{y}} \neq \mathbf{y}$$

1031
1032 Since both $\hat{\mathbf{y}}$ and \mathbf{y} are consistent with κ (which we assume satisfiable), we have:

$$1033 \quad \frac{\mathcal{P}(\hat{\mathbf{y}}|\mathbf{a}, \kappa)}{\mathcal{P}(\mathbf{y}|\mathbf{a}, \kappa)} = \frac{\mathcal{P}(\hat{\mathbf{y}}|\mathbf{a})}{\mathcal{P}(\mathbf{y}|\mathbf{a})}$$

1034
1035 Because $\hat{\mathbf{y}} = \mathcal{J}_{sc}(\mathbf{Y}, \kappa)(\mathbf{a}) = \arg \max_{\mathbf{y} \in \mathbb{B}^Y} \mathcal{P}(\mathbf{y}|\mathbf{a}, \kappa)$:

$$1036 \quad \mathcal{P}(\hat{\mathbf{y}}|\mathbf{a}, \kappa) \geq \mathcal{P}(\mathbf{y}|\mathbf{a}, \kappa)$$

1037
1038 Therefore:

$$1039 \quad \mathcal{P}(\hat{\mathbf{y}}|\mathbf{a}) \geq \mathcal{P}(\mathbf{y}|\mathbf{a}) \implies \mathbf{y} \neq \arg \max_{\mathbf{y} \in \mathbb{B}^Y} \mathcal{P}(\mathbf{y}|\mathbf{a}) = \mathbf{l}_{imc}(\mathbf{a})$$

1040
1041 Which is in contradiction with our premise, thus we have

$$1042 \quad \forall \mathbf{a} \in \mathbb{R}^k, \mathbf{y} \models \kappa, \mathbf{l}_{imc}(\mathbf{a}) = \mathbf{y} \implies \mathcal{J}_{sci}(\mathbf{Y}, \kappa)(\mathbf{a}) = \mathbf{y}$$

1043
1044 □

1045 B.5 RELATION BETWEEN SEMANTIC REGULARIZATION AND CONDITIONING

1046
1047 Finally, it is interesting to notice that under the consistent label hypothesis:

1048 **Proposition 6.**

$$1049 \quad \mathcal{L}_{sc}(\mathbf{Y}, \kappa)(\mathbf{a}, \mathbf{y}) = -\log(\mathcal{P}(\mathbf{y}|\mathbf{a})) + \log(\mathcal{P}(\kappa|\mathbf{a})) = \mathcal{L}_{sr}^{-1}(\mathbf{Y}, \kappa)(\mathbf{a}, \mathbf{y}) \quad (17)$$

1050
1051 *Proof 6.* Since labels are consistent, we have $\mathbb{1}_{\kappa}(\mathbb{1}_{\kappa}) = 1$, thus $\mathcal{P}(\mathbf{y}|\mathbf{a}, \kappa) = \frac{\mathcal{P}(\mathbf{y}|\mathbf{a})}{\mathcal{P}(\kappa|\mathbf{a})}$.

1052
1053 Therefore:

$$1054 \quad \begin{aligned} 1055 \quad \mathcal{L}_{sc}(\mathbf{Y}, \kappa)(\mathbf{a}, \mathbf{y}) &= -\log(\mathcal{P}(\mathbf{y}|\mathbf{a}, \kappa)) = -\log\left(\frac{\mathcal{P}(\mathbf{y}|\mathbf{a})}{\mathcal{P}(\kappa|\mathbf{a})}\right) \\ 1056 &= -\log(\mathcal{P}(\mathbf{y}|\mathbf{a})) + \log(\mathcal{P}(\kappa|\mathbf{a})) \\ 1057 &= \mathcal{L}_{sr}^{-1}(\mathbf{Y}, \kappa)(\mathbf{a}, \mathbf{y}) \end{aligned}$$

1058
1059 □

1060
1061 Thus, the loss module of semantic conditioning corresponds to that of semantic regularization with a $\lambda = -1$. Although it seems counter-intuitive that two systems trying to reach the same goal end up using “opposite regularization terms” in their loss module, this is justified by the different inference modules used in each system.



1062
1063 Hence, an implementation for $\mathcal{L}_{sr}^{\lambda}(\mathbf{Y}, \kappa)$ can be used for $\mathcal{L}_{sc}(\mathbf{Y}, \kappa)$. Besides, by training systems with regularized loss modules with different λ and evaluating with both $\mathcal{J}_{sc}(\mathbf{Y}, \kappa)$ and \mathbf{l}_{imc} , we can span the entire spectrum of techniques in *imc*, *sr*, *sc* and *sci*.

1064 C EXPERIMENTAL DETAILS

1065 C.1 TASKS

1066
1067 We give additional details on the tasks tackled in the paper.

1080 C.1.1 CATEGORICAL CLASSIFICATION

1081
1082 We mentioned earlier (see Section 3) how categorical classification tasks could be framed as a multi-
1083 label classification with prior knowledge. MNIST is one of the oldest and most popular dataset
1084 in computer vision and consists of small images of hand-written digits (e.g.  or ). Since its
1085 introduction in LeCun et al. (1998), it has been used as a *toy* dataset in many different settings.
1086 Likewise in neurosymbolic literature, many researchers used MNIST as a basis to build structured
1087 dataset compositionally (e.g. the PAIRS dataset in Marra et al. (2020), the MNIST-Add dataset in
1088 Manhaeve et al. (2021); Badreddine et al. (2022); Maene & De Raedt; van Krieken et al. or the
1089 Sudoku dataset in Augustine et al. (2022)).

1090 C.1.2 HIERARCHICAL CLASSIFICATION

1091
1092 The Cifar-100 dataset (Krizhevsky, 2009) is composed of 60,000 images classified into 20 mutually
1093 exclusive super-classes (e.g. *reptile*), each divided into 5 mutually exclusive fine-grained classes
1094 (e.g. *crocodile*, *dinosaur*, *lizard*, *turtle*, and *snake*). While most papers only consider the categorical
1095 classification task arising from the 100 fine-grained classes, we keep all 120 classes to produce a
1096 multi-label classification task where prior knowledge captures mutual exclusion and the hierarchy
1097 between super and fine-grained classes.

1098 The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2015) is an
1099 image classification challenge which has become a standard benchmark in computer vision to com-
1100 pare performances of deep learning models. As of August 2014, ImageNet contained 14,197,122
1101 annotated images organized into 21,841 synsets of the WordNet hierarchy (Miller, 1995), however
1102 standard image classification tasks often use a subset of those, usually 1,000 or 100 synsets. The
1103 WordNet hierarchy defines subsumption (or inclusion) between classes, and can be used in many
1104 ways to create a task of binary multi-label classification with prior knowledge.

1105 For our experiments, we sample 100 classes from 1k ImageNet and add all their parent classes.
1106 We then prune classes that have only one parent class and one child class to avoid classes having
1107 identical sample sets. We thus obtain a dataset of ImageNet samples labeled on a hierarchy of 271
1108 classes. Prior knowledge for this task includes the hierarchical knowledge coming from WordNet, as
1109 well as exclusion knowledge that we obtain by assuming two classes having no common descendants
1110 are mutually exclusive.

1111 C.1.3 SIMPLE PATH PREDICTION

1112
1113 The Warcraft shortest path task (Pogančić et al., 2019; Yang et al.; Niepert et al., 2021; Ahmed et al.,
1114 2022a) uses randomly generated images of terrain maps from the Warcraft II tileset. Maps are build
1115 on a 12×12 directed grid (each vertex is connected to all its *neighbors*) and to each vertex of the
1116 grid corresponds a tile of the tileset. Each tile is a RGB image that depicts a specific terrain, which
1117 has a fixed traveling cost. For each map, the label encodes the shortest s-t path (i.e., a path from
1118 the upper-left to the lower-right corners), where the weight of the path is the sum of the traveling
1119 costs of all terrains (i.e., grid vertices) on the path. The terrain costs are used to produce the dataset
1120 but are not provided during training nor inference. In the original dataset (Pogančić et al., 2019),
1121 output variables correspond to vertices in the grid and a state satisfies the simple path constraint if
1122 the vertices set to 1 constitute a simple s-t path.

1123 This representation comes with several issues. First, as noted in Ahmed et al. (2022a), the set of
1124 vertices ambiguously encode more than one path (because of cycles in the grid, there are several
1125 possible simple paths that go through the same vertices). Besides, computing MPE and PQE for
1126 simple path constraints on general directed graphs are respectively NP-hard (Karp) and #P-hard
1127 (Valiant). To make this task tractable, (Ahmed et al., 2022a) transforms the output space in the
1128 following way: edges of the grid are chosen as output variables instead of vertices and only simple
1129 paths with a maximal length of 29 (the maximal length found in the training set) are kept. This
1130 implies that the test set might not be consistent with the constraints since it might contain a path
1131 longer than 29 edges. Besides, such method would not scale to larger grids.

1132 In our experiments, we adopt a different approach. We keep the set of edges as our output vari-
1133 ables, but we turn the grid into an acyclic graph by only connecting vertices to their right and lower
neighbors. Acyclicity is a sufficient condition to compile simple path constraints to an Ordered Bi-

nary Decision Diagram (see `code/circuits/AcyclicSimplePath.py` in the supplementary material), which makes MPE and PQE tractable. This transformation allows us to scale to larger grids without an explosion of the computational cost. We recompute the labels for the new output space using the terrain costs.

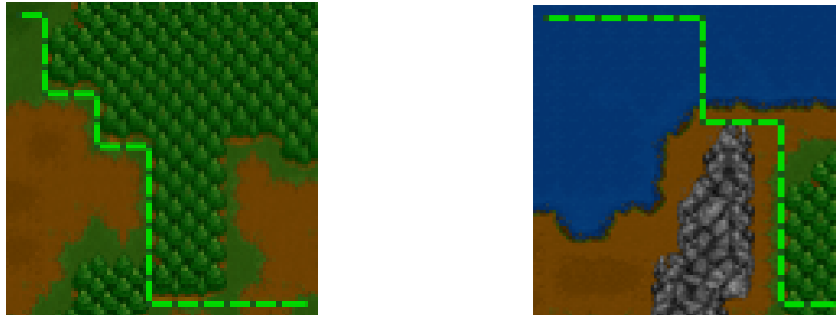


Figure 4: Examples of Warcraft maps and their shortest paths (Ahmed et al., 2022a)

C.2 IMPLEMENTATION

We used a simple Convolutional Neural Network (CNN) (LeCun et al., 1998) design on the MNIST dataset and the family of DenseNets (Huang et al., 2017) on all others.

In all our experiments, probabilistic reasoning computations brought no significant overhead on top of the neural network. For categorical tasks, the number of valid states is enumerable in linear time, hence there are no complexity issues to implement all three techniques. For hierarchical tasks, we implemented our own version¹ of (Deng et al., 2014), which uses a custom compilation algorithm to convert the propositional formula into a minimal junction tree and then applies a sparse message passing procedure. For simple path prediction task, we used the compilation technique mentioned above alongside the SPL package (Ahmed et al., 2022a) to implement semantic conditioning and semantic regularization. For semantic conditioning at inference, we simply adapted a shortest path solver from NetworkX (Hagberg et al., 2008) based on the Bellman-Ford algorithm (Bellman).

C.2.1 CATEGORICAL CLASSIFICATION

The architectural design for categorical classification on MNIST is a simple Convolutional Neural Network (CNN) (LeCun et al., 1998), as shown on Listing 1. We trained networks with `num_layers` from 1 up to 9 layers.

Listing 1: Our TinyNet architecture (PyTorch implementation)

```

class TinyCNNs(nn.Module):
    def __init__(self, num_classes: int = 10,
                 num_layers: int = 1,
                 in_channels=1):
        super().__init__()
        convs = []
        for i in range(num_layers):
            convs.append(nn.Conv2d(2**(i//2)*in_channels,
                                   2**((i+1)//2)*in_channels,
                                   5,
                                   padding=2))
            convs.append(nn.ReLU())
        self.convs = nn.Sequential(*convs)
        # self.AdaptativeScale = int(5*2**(num_layers/2))
        self.pool = nn.AdaptiveAvgPool2d(5)

```

¹Their code was not publicly available. Our code is attached in the supplementary materials and will be made publicly available in the final version.

```

1188         self.fc = nn.Linear(25*2**((num_layers//2)*in_channels,
1189                               num_classes)
1190
1191     def forward(self, x):
1192         x = self.convs(x)
1193         x = F.relu(x)
1194         x = self.pool(x)
1195         x = torch.flatten(x, start_dim=1)
1196         x = self.fc(x)
1197         return x

```

Then, to complete the neural based system: we use the loss module shown on Listing 2 (with varying λ) and inference modules shown in Listings 2 and 3 for *imc* and *sci* respectively.

Listing 2: *rsc* loss

```

1202 scores = self.model(x)
1203 energies = torch.gather(self.scores, 1, y.unsqueeze(dim=1))
1204 log_z = torch.sum(torch.log(torch.exp(self.scores).add(1)),
1205                   dim=1, keepdim=True)
1206 mc_log_z = torch.log(torch.sum(torch.exp(self.scores),
1207                               dim=1, keepdim=True))
1208
1209 loss = torch.mean(torch.sub(torch.add(log_z.mul(1+self.lambda),
1210                                   mc_log_z.mul(-self.lambda)),
1211                       energies))
1212
1213 return loss

```

Listing 3: *imc* inference

```

1216 scores = self.model(x)
1217 return torch.gt(scores, 0)

```

Listing 4: *sci* inference

```

1220 scores = self.model(x)
1221 _, idx_max = torch.max(self.scores, dim=1)
1222 return F.one_hot(idx_max, num_classes=scores.shape[1])

```

C.2.2 HIERARCHICAL CLASSIFICATION

The architectural design for hierarchical classification tasks Cifar and ImageNet was based on DenseNets (Huang et al., 2017). We used the `torchvision` implementation with a naive scaling strategy to create DenseNets of various size, as shown on Listing 5. We trained network with size from 0 up to 8.

Listing 5: DenseNet scaling

```

1231 from torchvision.models.densenet import _densenet
1232
1233 network = _densenet(growth_rate=32,
1234                    block_config=(6, 12, (size+3)*8, (size+1)*8),
1235                    num_init_features=64,
1236                    weights=None,
1237                    progress=True)

```

For the loss and inference modules, we followed (Deng et al., 2014) and expressed the hierarchical and exclusion relations as a HEX-graph H , then compiles this HEX-graph into a HEX-layer that can compute l_{κ_H} using a sparse max-product message passing algorithm with Viterbi decoding (see `fastHEXLayer.py`).

To do so, for ImageNet, we first extract hierarchical links from the `wn_hyp.pl` file and arrange them into a directed graph. Then, for our experiments, only 100 leaf nodes are randomly sampled from the total 1,000 and the directed graph is trimmed of any node not connected to a sampled leaf node. We also prune nodes that only have one parent and one child to avoid the case where two nodes have the same set of labeled samples (which would make them indistinguishable for the network). This directed graph is fed into a `HEXGraph` object, which computes the sparse and dense version of the hierarchical and exclusion matrices, builds the corresponding junction tree (using the `JunctionTree` object) and records the valid states of each clique and the sum-product matrix of the junction tree. The results of these compilation steps can be saved and loaded: the specific files used in this experiment are `./ImageNet/compilations/100p*`. During training, this `HEXGraph` is loaded from compilation files and passed on to the `HEXLayer` which contains the methods to perform *sci*. The code to perform those compilation steps is found in `ImageNetProcessing.ipynb`.

For Cifar-100, the hierarchy has only two levels (macro and fine-grained classes) and can be retrieved directly online and fed to the `HEXGraph` object.

C.2.3 SIMPLE PATH PREDICTION

The architectural design for simple path prediction on the Warcraft Shortest Path dataset is also based on DenseNets (Huang et al., 2017) as for hierarchical tasks. We trained network with `size` from 0 up to 6.

We compiled the constraints to an OBDD (Darwiche, 2009) using a custom algorithm. Compilation files can be found in the code under name files `12x12.sdd` and `12x12.vtree` in the `Code/WSP/data/12x12` folder.

Then, we used the SPL (Ahmed et al., 2022a) implementation to compute the loss module for semantic conditioning ($\lambda = -1$) and semantic regularization ($= 0.1$). For the inference module, we decided to replace the SPL implementation by an adapted shortest path solver from NetworkX (Hagberg et al., 2008) based on the Bellman-Ford algorithm (Bellman). We found this solution less prone to numerical stability issues.

C.3 METRICS

There are plenty of metrics that can be used to evaluate a classification system.

Simple accuracy averages how many classes were correctly labeled on each sample, however, since multi-label classification datasets with background knowledge are often highly unbalanced (far more negative classes than positive ones) it is often unfit to the task. Precision, recall and f1-score metrics can help tackle with this issue, but they lose track of the semantics of the task.

Semantic consistency counts how many outputs are consistent with the background knowledge. Since *sci* is provably consistent, this metric is of little interest for us.

Metrics that are not based on the binary outputs but need to access probability scores associated with each classe, like threshold metrics (e.g. `map@50`, `map@75`, `auc`) or top-k metrics, are not accessible to our classification system as is.

Eventually, we decide to use **exact accuracy**, which counts how many samples are perfectly labeled: this is the most demanding metric since a single mistake disqualifies the whole output. This metric is also used in Ahmed et al. (2022a) and in Xu et al. (2018) (where it is called coherent accuracy).

C.4 HYPERPARAMETERS

Epochs We trained each system on the training set for up to 100 epochs: 100 for MNIST, Cifar and Warcraft Shortest Path and only 90 for ImageNet due to computational resources constraints. We evaluate the perfect accuracy on the test set at each epoch.

Seeds We set seeds manually with `torch.manual_seed(args.seed)`. We used 6 seeds (`[0, 1, 2, 3, 4, 5]`) for MNIST, 3 seeds (`[0, 1, 2]`) for Cifar and ImageNet and 2 seeds (`[0, 1]`) for Warcraft Shortest Path (due to compute budget limits).

1296 **Batch size** We use a batch size of 8 for MNIST and Cifar, and increase to 64 for ImageNet and
1297 Warcraft Shortest Path to speed up training.
1298

1299 **Regularization coefficient** We used $\lambda = 0.1$ for semantic regularization (Xu et al., 2018). We did
1300 not perform a full hyperparameter search procedure because we considered it was too costly on large
1301 datasets like ImageNet, Cifar or Warcraft Shortest Path. We tried several values of λ on MNIST and
1302 at the beginning of our experiments on Cifar and ImageNet with no noticeable difference.
1303

1304 **Optimizer** We use Adam with a learning rate of 10^{-4} for all tasks.
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349