A Neuro-Symbolic Framework for Sequence Classification with Relational and Temporal Knowledge

Luca Salvatore Lorello[1,2*], **Marco Lippi**[3], **Stefano Melacci**[4]
[1]University of Pisa,
[2]University of Modena and Reggio Emilia,
[3]University of Florence,
[4]University of Siena
luca.lorello@phd.unipi.it, marco.lippi@unifi.it, stefano.melacci@unisi.it

Abstract

One of the goals of neuro-symbolic artificial intelligence is to exploit background knowledge to improve the performance of learning tasks. However, most of the existing frameworks focus on the simplified scenario where knowledge does not change over time and does not cover the temporal dimension. In this work we consider the much more challenging problem of knowledge-driven sequence classification where different portions of knowledge must be employed at different timesteps, and temporal relations are available. Our experimental evaluation compares multi-stage neuro-symbolic and neural-only architectures, and it is conducted on a newly-introduced benchmarking framework. Results demonstrate the challenging nature of this novel setting, and also highlight under-explored shortcomings of neuro-symbolic methods, representing a precious reference for future research.

1 Introduction

Sequence classification is a very well-known task in machine learning which can become very challenging when decisions must be made on top of complex features or without suitable priors, especially when safety, reliability and accountability are of paramount importance (autonomous driving [Roesener *et al.*, 2016], industrial control systems [Chakraborty *et al.*, 2022], medicine [Ivaturi *et al.*, 2021], and others). The task can be also approached by symbolic methods exploiting grounded frameworks, such as regular expressions [Galassi and Giordana, 2005] and automata induction [Angluin, 1982], or temporal logics satisfiability [Rozier and Vardi, 2007]. However, they tend to struggle in the presence of noise [Umili and Capobianco, 2024], and may be unsuitable for open-world settings where background knowledge or inductive priors are not available at all. Neuro-symbolic artificial intelligence [Besold *et al.*, 2021] aims to get the best of both worlds, by merging neural and symbolic methods, as in DeepProbLog [Manhaeve *et al.*, 2018], or Logic Tensor Networks [Badreddine *et al.*, 2022]. Little has been done for sequence classification in presence of (i) "relational" and (ii) "temporal" knowledge. The former describes features within each single time-step while the latter models information along the temporal dimension. For example, consider a safety-critical system where several cameras monitor an environment, controlling the movements of automated guided vehicles. At each timestep, every vehicle has to satisfy some safety properties regarding its movements, for example to avoid collisions; in addition, at some checkpoints it is necessary to assess that some tasks have been executed in a specific order throughout the temporal sequence. Motivated by this challenging and still under-explored setting, this paper collects several contributions. (a) We propose a new benchmarking framework for sequence classification (and similar tasks) in the presence of domain knowledge that extends also over the temporal dimension. Our framework is capable of generating datasets with multi-channel sequences of arbitrary length, by sampling given image classification datasets according to user-defined temporal and relational specifications. (b) We share with the scientific community ready-to-use datasets for six challenging tasks, as well as baseline performances for neural-only and modular neuro-symbolic architectures. Building on top of a neural net processing the raw data, our neuro-symbolic architecture consists of an automata-based temporal reasoner stacked on top of a relational symbolic module, both representatives of state of the art approaches in their respective categories. (c) Our detailed experimental study highlights two under-investigated problems: specific methods for temporal reasoning, highly-effective with propositional inputs, struggle when extended to a relational setting, even when the interface is "propositionalized" by an upstream reasoner, and, conversely, relational neuro-symbolic methods present training instabilities when coupled with downstream (temporal) reasoning components.

2 Background

Linear temporal logic over finite traces. A dynamic discrete system is a triple $\langle S, R, s_0 \rangle$, where S is a set of (possibly infinite) states, R is a transition relation between states, often non-deterministic, and $s_0 \in S$ is the initial state. An execution *trace* is a (possibly infinite) sequence of states $(s_0, s_1, s_2, \dots)$, such that $R(s_{i-1}, s_i), \forall i > 0$. Linear Temporal Logic (LTL) [Pnueli, 1977] is a formalism for tempo-

*Part of the work was done while LSL was visiting KU Leuven. Supplementary material at: https://arxiv.org/abs/2505.05106

ral reasoning over infinite execution traces of dynamic discrete systems, capable of expressing properties related to state reachability. Given a sub-formula or an atomic proposition, indicated with ϕ or ψ in the following, the *next* operator $\bigcirc$ models immediate future reachability, i.e., $\bigcirc\phi$ means that ϕ has to hold at the next state. Other operators model eventual reachability (*finally/eventually* operator, $\Diamond\phi$, i.e., ϕ has to hold at least once in the future), invariance (*globally* operator, $\Box\phi$, i.e., ϕ has to hold on the entire subsequent path) and conditional reachability (such as the binary operators *until*, $\phi\mathcal{U}\psi$, and *releases*, $\phi\mathcal{R}\psi$). The original formulation of LTL can modified in multiple ways: LTL over finite traces (LTL_f) [De Giacomo and Vardi, 2013] is a restriction which preserves the same syntax of LTL[1] but redefines its semantics, in order to be applied to execution traces of finite length. This restriction also has practical repercussions: while LTL can express ω-languages recognized by Büchi automata, reasoning over LTL_f can be reduced to regular languages, accepted by deterministic finite state automata (DFA). In this work we convert LTL_f formulas to DFAs, using flloat[2].

Neuro-symbolic integration of logic formulas. The main challenge of neuro-symbolic integration [Besold *et al.*, 2021] consists in providing an interface between two components: learning by means of neural networks, which requires representations in a continuous space, and reasoning, which often benefits from discrete representations of symbols. A popular approach for the integration of logic knowledge, known as the model-theoretic approach [Marra *et al.*, 2024], is to relax truth assignments, in a way which extends Boolean algebra in a continuous and differentiable space. Fuzzy logic [Badreddine *et al.*, 2022] extends interpretations in the continuous range $[0, 1]$ and replaces Boolean conjunction with a t-norm.[3] The obtained expressions are equivalent to the original Boolean formula at boundary values, but allow differentiability by means of a progressive transition between truth values, which can constrain the learning procedure [Gnecco *et al.*, 2015]. Different choices of t-norms are possible, each characterized by different advantages and drawbacks. An alternative framework for model-theoretic neuro-symbolic integration, overcoming some issues with differentiable fuzzy logics [van Krieken *et al.*, 2022], is based on probabilistic inference. In this framework, with strong theoretical and computational foundations in statistical and relational artificial intelligence [De Raedt *et al.*, 2016], Boolean propositions are seen as Bernoulli random variables and logic connectives are interpreted as set operators. Weighted model counting (WMC) is a general framework for probabilistic inference. Algebraic model counting (AMC) [Kimmig *et al.*, 2017] extends WMC by replacing Boolean operators with elements of an algebraic semiring, allowing to solve a plethora of probabilistic tasks within a single framework. WMC and AMC are in general #P-COMPLETE [Chavira and Darwiche, 2008], however they become tractable when logic formulas possess a specific structure. Knowledge compilation [Darwiche and Marquis, 2002] amortizes execution time by converting input formulas into equivalent target normal forms, on top of which clausal inference (and possibly other classes of tasks) can be executed in polynomial time. In this context, we mention sd-DNNF, the smooth and decomposable deterministic negated normal form, a popular knowledge compilation target language, which guarantees correct model counting in polynomial time, also in the presence of neutral and disjoint sums.

[1] LTL_f, introduces an additional operator, weak next, $\mathcal{W}\phi$, to deal with last state semantics in finite traces.

[2] https://github.com/whitemech/flloat

[3] In propositional logic, every other operator is constructed by exploiting the definition $\neg p = 1 - p$ and equivalence axioms.

3 Related Works

Neuro-symbolic temporal reasoning. Neuro-symbolic temporal reasoning has been employed for temporal formula induction, approximate satisfiability and sequence classification driven by background knowledge. [Camacho and McIlraith, 2019] address formula induction in a symbolic fashion, by building a vocabulary of subformulas which are converted to alternating finite automata, and then composed to discover the target formula from positive and negative examples, while [Walke *et al.*, 2021] propose the use of specialized recurrent layers for sequence classification, which can collectively be interpreted as an LTL_f formula, by extracting a truth table from discretized weights. Finally, [Umili and Capobianco, 2024] attempt to learn DFA transition matrices from examples, by discretizing a recurrent neural network, regularized during training to produce discrete activations. Exact LTL satisfiability is PSPACE-COMPLETE. However, recently, several approaches based on neural networks have been proposed to approximate satisfiability in polynomial time: [Xie *et al.*, 2021] employ message passing graph neural networks to learn embeddings for a DFA equivalent to the target LTL_f formula; [Mukherjee *et al.*, 2022] exploit graph isomorphism networks to perform approximate model checking; [Luo *et al.*, 2022; Luo *et al.*, 2024] use recursive neural networks to generate explanations as traces satisfying the given formulas.

Neuro-symbolic finite state automata. There is a strong link between inference in DFA and recurrent neural networks, as the former performs trace acceptance using a state transition matrix and the latter updates an internal state according to a deterministic function. On the other hand, the discrete nature of DFAs is particularly amenable to be encoded by (sets of) Boolean formulas, which, in turn, can be subject to neuro-symbolic integration. Starting from this consideration, [Umili *et al.*, 2023] exploit background knowledge in the form of LTL_f formulas to learn symbolic labels from sequence labels, by performing distant supervision [Manhaeve *et al.*, 2018]. The transition matrix is converted into a disjunction of Horn clauses and the entire logic program is finally encoded into a Logic Tensor Network [Badreddine *et al.*, 2022] that can be trained end-to-end on sequence labels, by means of binary cross-entropy or a proposed semantic loss penalizing discordance between ground truth labels and the observed final state. [Manginas *et al.*, 2024] highlight the shortcomings of this method, by identifying a general failure on more complex datasets, consisting of larger DFAs, longer temporal horizons or more complex transition guards, and pinpointing the cause on the fuzzy encoding. In their extension, each tran-

sition guard is compiled into an sd-DNNF, and the automaton is given probabilistic semantics, by means of AMC on the probability semiring, largely improving scalability.

Relationships to this paper. We take inspiration both from [Umili *et al.*, 2023] and [Manginas *et al.*, 2024], however we allow transition labels to contain constraints (i.e., first order predicates of known semantics, applied to terms which represent objects in finite domains). This distinction places our work closer to symbolic automata [Veanes *et al.*, 2010; Veanes, 2013; D'Antoni and Veanes, 2017], and it is characterized by additional challenges. We address these challenges in a straight-forward manner, by decoupling reasoning in two components: relational and temporal inference. In this way, temporal inference is reduced to the propositional case, decoupling the additional challenges provided by a first order setting, from those posed by more complex temporal behaviors. From a neuro-symbolic perspective, our decoupling approach also provides benefits in terms of knowledge injection: as the semantics of states is defined over constraint satisfaction problems, background knowledge is richer and more structured (i.e., constraints act as relations between concepts, while propositional formalisms are limited to assignments).

4 Sequence Classification with Relational and Temporal Knowledge

Problem definition. Let $\mathcal{S} = ([x_0^t, \ldots, x_{N-1}^t])_{t=0}^{T-1}$ be a sequence of length T that consists, for each time step t, of N perceptual stimuli belonging to as many *perceptual domains* $\mathcal{X}_j$: $x_j^t \in \mathcal{X}_j$. Each x_j^t is associated to a symbol y_j^t that belongs to a *symbolic domain* $y_j^t \in \mathcal{Y}_j$. As an example, $\mathcal{X}_0$ and $\mathcal{X}_1$ could be the domain of MNIST digits [LeCun, 1998] and Fashion-MNIST articles [Xiao *et al.*, 2017], respectively, whereas $\mathcal{Y}_0$ and $\mathcal{Y}_1$ the corresponding sets of (symbolic) classes. To avoid confusing the stimulus/domain index with the time index, we will sometimes use letter subscripts in place of numbers (e.g., A, B in place of 0, 1) to refer to different stimuli, symbols, and domains (e.g., x_A^t, x_B^t, y_A^t, y_B^t, $\mathcal{X}_A$, $\mathcal{X}_B$, $\mathcal{Y}_A$, $\mathcal{Y}_B$, etc.). We consider the problem of learning a binary classifier $f\colon \mathcal{S} \mapsto \{0,1\}$ which maps the input sequence to the positive class if and only if $\mathcal{S} \models \mathcal{T}$, where $\mathcal{T}$ is a temporal specification that consists in an LTL$_f$ formula whose atomic symbols are first order relations grounded over *symbolic domains* $\mathcal{Y}_j$. Intuitively, this means that the sequence classification task consists in assessing whether two sets of properties jointly hold or not: (i) instantaneous relations among the N symbolic domains which may or may not hold for any t ("relational"), and (ii) meta-relations about the validity of each instantaneous property over time ("temporal"). The reasoning process in such setting can be reduced to querying the validity of each instantaneous property to build a trace of truth values, and then checking whether the trace satisfies the temporal specification. In order to preserve LTL$_f$ conciseness and readability, we encode "relational" (i.e., instantaneous) properties as constraints over finite domains [Nethercote *et al.*, 2007], which allow to express arithmetic, identity-based, ordering-based, lexicographic, etc., relations with well-codified, compact, and efficient to compute constructs, known as global constraints. To further aid readability, we collect (i) relational properties into a set of constraints $\mathcal{C}$, separately from the (ii) "temporal" formula in $\mathcal{F}$. It is important to note that, in our setting, it is the joint effect of the LTL$_f$ formula and the properties with which its symbols are grounded (i.e., the complete background knowledge is $\mathcal{T} = \mathcal{C} \cup \mathcal{F}$), which acts as decision rule. As an example, let us consider two popular digit classification datasets, MNIST and SVHN [Netzer *et al.*, 2011], where class names can be also interpreted as class indices. Suppose that we have a sequence $\mathcal{S}$ where, at each time instant, we are given $N = 3$ stimuli (x_A^t, x_B^t, x_C^t), $x_A^t \in \mathcal{X}_A$, $x_B^t \in \mathcal{X}_B$, with $\mathcal{X}_A = \mathcal{X}_B =$ MNIST digits, while $x_C^t \in \mathcal{X}_C =$ SVHN digits. Let us consider the case in which the N perceptual stimuli are mapped to N symbols (y_A^t, y_B^t, y_C^t), belonging to domains $\mathcal{Y}_A = \mathcal{Y}_B = [0, 9]$, and $\mathcal{Y}_C = [2, 8]$ (integer). As "relational" knowledge we assume that two properties can potentially hold: $y_A + y_B = y_C$ and $y_A \neq y_B \wedge y_B \neq y_C \wedge y_C \neq y_A$,[4] where we compactly indicate the latter with $\texttt{all\_different}(y_A, y_B, y_C)$. We then introduce some "temporal" knowledge: in even timesteps the sum constraint is expected to hold, while `all_different` is expected to hold for odd steps. The goal is to learn a binary classifier, distinguishing knowledge-coherent sequences from incoherent ones. The properties of our problem can more formally and concisely be defined as the quadruple $\langle \mathcal{X}, \mathcal{Y}, \mathcal{C}, \mathcal{F} \rangle$ where $\mathcal{X}$ and $\mathcal{Y}$ are the unions of all the perceptive and symbolic domains, respectively; $\mathcal{C}$ is the set encoding the relational knowledge; $\mathcal{F}$ is the temporal formula. We have:

$$
\begin{aligned}
&\mathcal{X}\colon && \mathcal{X}_A, \mathcal{X}_B = \{\,\texttt{0}, \texttt{1}, \ldots, \texttt{9}\,\}; && \mathcal{X}_C = \{\,\texttt{2}, \texttt{3}, \ldots, \texttt{8}\,\}\\
&\mathcal{Y}\colon && \mathcal{Y}_A, \mathcal{Y}_B = [0, 9]; && \mathcal{Y}_C = [2, 8]\\
&\mathcal{C}\colon && p : y_A + y_B = y_C\\
& && q : \texttt{all\_different}(y_A, y_B, y_C)\\
&\mathcal{F}\colon && p \wedge \Box(p \leftrightarrow \bigcirc q).
\end{aligned}
$$

Therefore, a sequence of triples such as $([\texttt{0}, \texttt{8}, \texttt{8}], [\texttt{1}, \texttt{7}, \texttt{3}], [\texttt{3}, \texttt{2}, \texttt{5}], [\texttt{2}, \texttt{4}, \texttt{8}])$, corresponding to the trace $([p, \neg q], [\neg p, q], [p, q], [\neg p, q])$, will be accepted by a knowledge-driven sequence classifier, while $([\texttt{0}, \texttt{8}, \texttt{8}], [\texttt{1}, \texttt{7}, \texttt{3}], [\texttt{3}, \texttt{1}, \texttt{5}], [\texttt{2}, \texttt{4}, \texttt{8}])$, corresponding to the trace $([p, \neg q], [\neg p, q], [\neg \mathbf{p}, q], [\neg p, q])$ will not, as $\neg \mathbf{p}$ (bold) violates the temporal property.

The LTLZinc Framework. LTLZinc[5] is a benchmarking framework for temporal reasoning tasks, capable of generating sequences from user-defined constraint specifications, involving relational and temporal knowledge. LTLZinc receives as input the quadruple $\langle \mathcal{X}, \mathcal{Y}, \mathcal{C}, \mathcal{F} \rangle$, as well as additional options, and produces an output depending on either of two modes: (i) sequential and (ii) incremental. In (i) *sequential* (which could informally be categorized as "learning about time"), the output will be a dataset of densely annotated sequences, which can be used for binary classification, and other temporal-related tasks. In (ii) *incremental* mode (which could informally be classified as "learning over

[4] In MiniZinc [Nethercote *et al.*, 2007] enumerations are integers, allowing arithmetic expressions also for categorical symbols.

[5] https://github.com/continual-nesy/LTLZinc

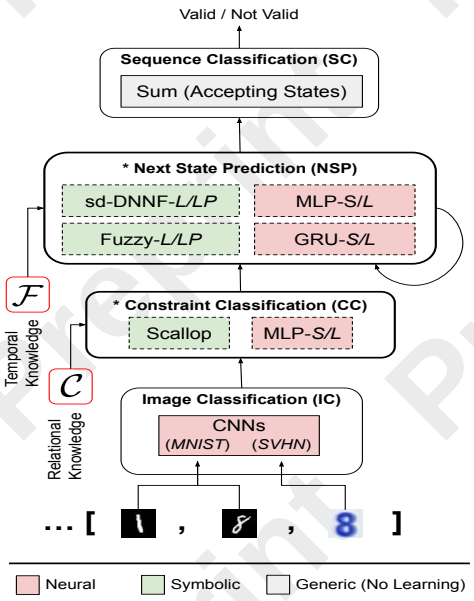

Figure 1: Stages of our architecture. CC and NSP are instantiated in multiple ways, shown as green (symbolic) and red (neural) blocks.

time"), LTLZinc outputs a single sequence of datasets, to be used in incremental/continual [Wang *et al.*, 2024] and curriculum learning settings. As LTL is strictly more expressive than current definitions of incremental learning in the literature, it is straight-forward to encode class-incremental, task-incremental and domain adaptation settings with LTLZinc. Internally, LTLZinc converts the temporal specification $\mathcal{F}$ into a DFA annotated with constraints, and solves a collection of MiniZinc programs, corresponding to the subset of $\mathcal{C}$ satisfying each of the transition guards, caching every solution found. Then, (positive/negative) sequences of user-specified length are generated by random walks along the automaton, where at each transition one solution of the corresponding problem is randomly sampled. Annotations contain the background knowledge, consisting of the user specification (the quadruple $\langle \mathcal{X}, \mathcal{Y}, \mathcal{C}, \mathcal{F} \rangle$) and the generated automaton, sequence-level binary labels, the trace of traversed states during generation, the trace of constraint truth values, and the trace of image labels for each stimulus of the N-uple).

5 Methodology

We model the whole sequence classification task by a multi-stage pipeline composed of the following sub-tasks, also sketched in Fig. 1: (IC) image classification, mapping data from each $\mathcal{X}_i$ to the corresponding $\mathcal{Y}_i$; (CC) constraint classification, leveraging "relational" knowledge; (NSP) next state prediction, leveraging "temporal" knowledge; (SC) sequence classification, i.e., the binary classification problem. LTLZinc provides annotations for each of these sub-tasks, so that we can evaluate performance both in isolation for each component, and jointly in an end-to-end fashion, while also being able to define multiple training objectives, applied at different stages. We address the following research questions.

Q1: Can neural-only methods solve the proposed tasks? Given enough training capacity, can a neural network successfully perform sequence classification, without background knowledge?
Q2: How do neural-only and neuro-symbolic methods compare? Can background knowledge mitigate error amplification issues along multiple stages? Are neuro-symbolic methods more prone to training instability than neural-only approaches?
Q3: What is the effect of upstream noise on exact symbolic methods? Do wrong, but confident, predictions of symbols y_i's affect downstream accuracy more than correct predictions associated with lower confidence? How is noise propagated?

(IC) **Image classification.** The first stage corresponds to a traditional neural-based image classification task, estimating the probability of the class assignment $[y_0^t, \ldots, y_{N-1}^t]$, i.e., $P_{\text{IC}}([y_0^t, \ldots, y_{N-1}^t] \mid [x_0^t, \ldots, x_{N-1}^t])$. Our implementation is based on a convolutional architecture (details in Appendix B.1), for each perceptual domain (e.g., for the example of Section 4, we have two instances of the same architecture: one to predict symbols y_A, y_B (MNIST categories) and another one for symbols y_C (SVHN categories), see Fig. 1).

(CC) **Constraint classification.** The set of all the image classes predicted by the IC module, is mapped to $|\mathcal{C}|$ validity values, each of them indicated with β_i, one for each of the constraints defined in the relational knowledge $\mathcal{C}$. This stage estimates $P_{\text{CC}}([\beta_0^t, \ldots, \beta_{|\mathcal{C}|-1}^t] \mid [y_0^t, \ldots, y_{N-1}^t])$. Our experiments focus on tasks characterized by constraints which can be expressed in Datalog. Hence, we chose Scallop [Li *et al.*, 2023], a neuro-symbolic engine capable of probabilistic reasoning over Datalog programs, supporting inference over multiple provenance semirings [Green *et al.*, 2007].[6] Scallop programs are differentiable end-to-end, however they do not possess trainable parameters. To increase module flexibility, we augment our architecture with an optional set of additional learnable calibration parameters (enabled by a "calibrate" hyper-parameter in the experiments of Section 6), which independently apply temperature rescaling to both input and output probabilities. Learning takes place jointly with the IC module, and it allows us to control both the entropy of the distributions and the confidence of prediction, while preserving the argmax. We compare Scallop with two fully-learning-based approaches: a small multi-layer perceptron with limited capacity (8 hidden neurons, MLP-S) and a larger one (64 hidden neurons, MLP-L), see Appendix B.2.

(NSP) **Next state prediction.** The temporal reasoning component is rooted on the definition of a discrete space of M elements, each consisting of a state of the DFA equivalent to $\mathcal{F}$, based on the observed validity values of relational constraints, β_i^t's. In this way, temporal reasoning is reduced to to a next-state prediction problem, i.e., a recurrent classification problem where $\alpha^t \in [0, M-1]$ is the predicted class/next-state, with probability $P_{\text{NSP}}(\alpha^t \mid [\beta_0^t, \ldots, \beta_{\mathcal{V}-1}^t], \alpha^{t-1})$.[7] We compare four learning-based approaches (two MLP's, namely MLP-S and MLP-L, and two gated recurrent units, GRU-S, GRU-L, with 8 and 64 hidden/state neurons, respectively),

[6] In initial explorations, the default top-k proofs provenance ($k = 1$) resulted in the best trade-off between inference time and accuracy.

[7] Here and in the following, $t = -1$ is the initial instant, where the automaton of the NSP module is initialized to state "0".

Task	Config. CC-NSP	Best Model CC NSP (epoch)	Avg Acc. ↑	IC acc. ↑	CC Acc. ↑	NSP Acc. ↑	SC Acc. ↑
Task 1	Neural-Neural	MLP-L MLP-L (24)	0.84	**0.88**	0.85 ± 0.01	0.72	0.90
Task 1	Neural-Symbolic	MLP-L sd-DNNF-LP (24)*	0.85	0.86	0.85	0.81	0.90
Task 1	Symbolic-Neural	Scallop MLP-L (16)†	0.82 ± 0.07	0.88	0.91	0.59 ± 0.26	0.90
Task 1	Symbolic-Symbolic	Scallop Fuzzy-P (24)†*	**0.88**	0.87	0.91 ± 0.01	**0.84** ± 0.01	**0.91** ± 0.01
Task 2	Neural-Neural	MLP-S GRU-L (23)†	0.69 ± 0.03	0.88 ± 0.01	0.87 ± 0.01	0.50 ± 0.13	0.50
Task 2	Neural-Symbolic	MLP-S Fuzzy-LP (24)†	0.71	0.81 ± 0.03	0.82 ± 0.02	0.56 ± 0.02	0.67 ± 0.03
Task 2	Symbolic-Neural	Scallop GRU-L (22)*	0.72 ± 0.03	0.90	**0.93**	0.56 ± 0.12	0.50
Task 2	Symbolic-Symbolic	Scallop Fuzzy-P (16)†*	**0.79** ± 0.01	**0.89** ± 0.01	0.91	**0.70** ± 0.01	**0.67** ± 0.04
Task 3	Neural-Neural	MLP-L GRU-L (24)	0.68	0.96	0.82 ± 0.01	0.44	0.50
Task 3	Neural-Symbolic	MLP-L sd-DNNF-P (24)*	0.66 ± 0.01	0.93	0.71 ± 0.02	**0.51** ± 0.03	0.49 ± 0.01
Task 3	Symbolic-Neural	Scallop GRU-L (19)†	**0.73** ± 0.01	**0.98**	**0.98** ± 0.01	0.45 ± 0.03	0.50
Task 3	Symbolic-Symbolic	Scallop sd-DNNF-LP (20)*	0.55 ± 0.36	0.43 ± 0.47	0.64 ± 0.28	0.49 ± 0.46	**0.63** ± 0.28
Task 3	(Symbolic-Symbolic)	Scallop sd-DNNF-LP (20)*	0.93	0.96	0.96	0.90	0.91
Task 4	Neural-Neural	MLP-L GRU-S (21)†*	0.62 ± 0.06	**0.89** ± 0.01	0.80 ± 0.01	0.30 ± 0.26	0.50
Task 4	Neural-Symbolic	MLP-S sd-DNNF-P (19)†	0.60 ± 0.06	0.80	0.62 ± 0.04	0.49 ± 0.11	0.48 ± 0.12
Task 4	Symbolic-Neural	Scallop GRU-L (20)†*	0.68 ± 0.01	0.88 ± 0.01	**0.86** ± 0.02	0.46 ± 0.02	0.50
Task 4	Symbolic-Symbolic	Scallop sd-DNNF-P (16)†*	**0.74** ± 0.05	0.83 ± 0.01	0.79 ± 0.01	**0.69** ± 0.08	**0.67** ± 0.09
Task 5	Neural-Neural	MLP-L MLP-L (24)*	0.61 ± 0.01	0.95 ± 0.01	0.57 ± 0.04	0.42 ± 0.03	0.50
Task 5	Neural-Symbolic	MLP-L sd-DNNF-P (4)	0.54 ± 0.01	0.94 ± 0.01	0.50 ± 0.05	0.23	0.50
Task 5	Symbolic-Neural	Scallop MLP-L (24)†*	**0.88** ± 0.06	**0.98**	**0.96**	**0.78** ± 0.02	**0.80** ± 0.26
Task 5	Symbolic-Symbolic	Scallop Fuzzy-P (23)	0.85 ± 0.01	0.98	0.96	0.74 ± 0.02	0.71 ± 0.05
Task 6	Neural-Neural	MLP-L MLP-L (22)	0.67 ± 0.01	0.96	0.74 ± 0.01	0.48 ± 0.10	0.48 ± 0.06
Task 6	Neural-Symbolic	MLP-S Fuzzy-P (22)	0.56 ± 0.09	0.81 ± 0.15	0.53 ± 0.12	0.39 ± 0.06	0.52 ± 0.03
Task 6	Symbolic-Neural	Scallop GRU-L (16)†	0.80 ± 0.07	**0.98**	**0.98** ± 0.01	0.63 ± 0.18	0.62 ± 0.11
Task 6	Symbolic-Symbolic	Scallop sd-DNNF-P (21)†*	**0.85** ± 0.05	0.97	0.97	**0.70** ± 0.08	**0.76** ± 0.10

Table 1: **Q2**. Test set accuracies (mean ± std–if non-zero–, 3 runs), for different configurations ("Neural-Symbolic" = Neural CC, Symbolic NSP). Best model (named "CC model NSP model") and epoch selected by Avg Acc. on validation set. * = semantic loss; † = calibrated.

and four symbolic approaches. GRUs are augmented with a simple encoder-decoder, to convert the continuous hidden state into M discrete classes. For the symbolic approaches, we encode the ground truth DFA, following [Umili *et al.*, 2023] and [Manginas *et al.*, 2024]. Such automaton is a set of propositional formulas as PREV_STATE $\wedge$ TRANS_LABEL $\rightarrow$ NEXT_STATE, encoded by logic tensor networks [Badreddine *et al.*, 2022] (here named Fuzzy) and sd-DNNF [Darwiche and Marquis, 2002], respectively. To assess the effect of numerical stability, we performed computations both in probability space and log-probability space (suffixes -P and -LP, respectively). As in the CC module, automata-based NSP's predictions are optionally temperature-calibrated.

(SC) **Sequence classification.** The final output α^{T-1} of NSP directly encodes the final state of the automaton. Hence, we can perform sequence classification in closed form: $P_{\text{SC}}(f(\mathcal{S}) = 1 \mid \alpha^{T-1}) = \sum_{s \in \text{ACCEPTING}} P(\alpha^{T-1} = s)$.

Training the pipeline. The four stages are combined into: $P(f(\mathcal{S}) = 1) = \left(\prod_{t=0}^{T-1} P^t_{\text{IC}} P^t_{\text{CC}} P^t_{\text{NSP}} \right) P_{\text{SC}}(f(\mathcal{S}) = 1 \mid \alpha^{T-1})$, where $P^t_{\cdot}$ is the shorthand notation for the already introduced probabilities with arguments at time t. We train our multi-stage architectures with four loss functions, each weighted by a $\lambda_{\cdot}$ hyper-parameter. IC and NSP (the latter conditioned on the previous state) exploit a categorical cross-entropy loss, while CC and SC a binary cross-entropy loss. For SC, we also evaluate the semantic loss proposed by [Umili *et al.*, 2023]. Preliminary experiments demonstrated that training diverges due to extremely low initial confidence in image classification,making the optimizer unable to converge, except in very simple tasks. This behavior is well known in the literature [Manhaeve *et al.*, 2021; van Krieken *et al.*, 2024; Maene *et al.*, 2024], and, in this paper, it only affects experiments aimed at investigating **Q2**, where we bootstrap the IC module with a 5 epochs pre-training phase, using the IC-loss only, to ensure a good starting image classification.

6 Experiments

We hereby present an experimental evaluation conducted on six LTLZinc tasks.[8] Each consists in a binary classification problem on 400 image sequences of variable length (between 10 and 20), with images sampled from MNIST [LeCun, 1998] and Fashion MNIST [Xiao *et al.*, 2017]. Tasks are annotated with relational and temporal knowledge (see Appendix A for all the details). We address research questions of Section 5 with three distinct experimental activities, aimed to study the effect of different architectures and available knowledge. Notice that our goal is not to simply find the best performing model, but to explore different scenarios and gain insights on our challenging setting. Appendices B.4, B.5 and B.6 report values of every hyper-parameter.

Q1. To explore neural-only methods, our modular architecture is instantiated with neural components only (i.e., the convolutional backbone (IC), MLP-S/L for constraint prediction (CC), and MLP-S/L or GRU-S/L for next-state prediction (NSP), Fig. 1). Supervision is provided at every level, weighting each loss by the same positive coefficient ($\lambda_{\cdot} = 1.0$). Our expectations were that the neural-only pipeline would have been good in managing the IC stage, but it was not clear how it would have performed in the other ones. Results (detailed in Appendix C) confirmed that, despite their perceptual simplicity, no task can be effectively learned with sequence-level labels alone. We experienced slow convergence and sub-optimal performance at the end of training. In fact, in

[8] LTLZinc generator, our dataset, and code for full reproducibility: https://github.com/continual-nesy/LTLZinc.

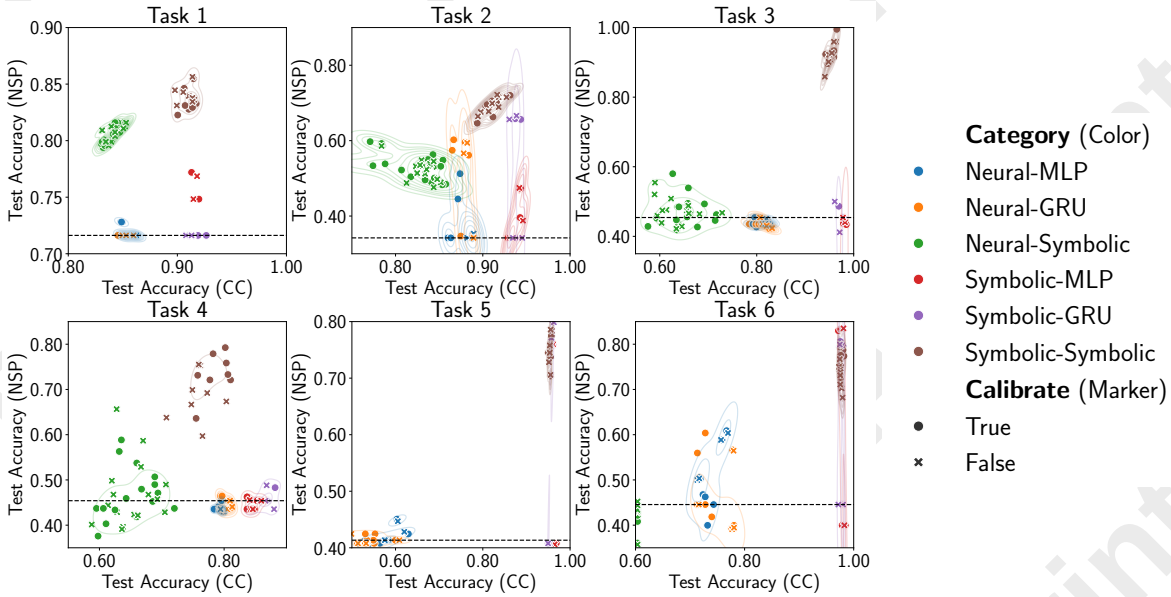

Figure 2: **Q2**. CC-NSP accuracy trade-off for different families of architectures (i.e., Neural/Symbolic CC, MLP/GRU/Symbolic NSP). The dashed line indicates the baseline performance of a deterministic NSP always selecting the state most represented in the training set.

spite of good overall image classification, the optimizer is often stuck in local equilibrium points, causing performance for other training objectives to plateau multiple times during training. Harder reasoning tasks exacerbate this effect.

Q2. This second experimental batch aims to compare neural vs. neuro-symbolic methods, thus the effect of background knowledge over multiple reasoning steps, as well as the interaction between components. We fix a neural perceptual backbone (IC), and then build four categories by combining either Neural (red, Fig. 1) or Symbolic (green, Fig. 1) modules for constraint prediction (CC) and temporal reasoning (NSP). We will indicate each configuration with a shorthand notation, e.g., Symbolic-Neural means Symbolic CC and Neural NSP modules. The use of the semantic loss and/or of calibration (Section 5), are treated as additional hyperparameters. Training is performed in two steps: 5 pre-training epochs for the IC module, and then 20 epochs of training for the entire architecture. Overall, these experiments confront 96 combinations, across 6 tasks, each seeded 3 times.

Q3. The third batch of experiments focuses on symbolic-only methods, thus on the Symbolic-Symbolic (green, Fig. 1) architectures. We exploit probability calibration (Section 5) as the only form of learning, to minimize noise and precisely pinpoint the effect of variables such as upstream label uncertainty, and supervision "distance" from the reasoning component. We also consider replacing IC and CC modules with specific "oracles" returning the ground truth labels, possibly with some level of corruption. In particular, we either replace only IC or both IC and CC. An oracle is characterized by two hyper-parameters: how ground truth is corrupted before feeding the next module, and the amount of corruption in terms of noise probability p. "Flip oracles" return correct labels with probability $1 - p$, and random labels with probability p, both with confidence 1.0. "Confidence oracles", on the other hand, always return correct labels with random confidence between $1 - p$ and 1.0, redistributing the remaining mass to other labels. When $p = 0.0$, both oracles yield ground truth labels with maximum confidence, referred to as "perfect oracle".

6.1 The Impact of Neuro-Symbolic Approaches

Fig. 2 highlights the trade-offs between CC and NSP, as a function of the different Neural/Symbolic implementations. Exploiting a symbolic component both for CC and NSP (Symbolic-Symbolic, brown points) allows to achieve the best trade-off for every task considered. Unlike other combinations, these approaches consistently outperform naive baselines which always return the most probable class observed in the training set, both for NSP (dashed horizontal line), and CC (not shown, outside the left boundary of plots). The additional learning capacity provided by temperature calibration (dots vs. crosses) has an overall limited effect. With the exception of tasks 1, 2 and 4, neural modules for NSP (clusters Symbolic-MLP/GRU and Neural-MLP/GRU, blue, orange, red and purple) achieve unsatisfactory performance, even when fed from highly-accurate symbolic CC predictions. These architectures, at times, perform on par with the most probable guessing baseline, and often below it. Symbolic modules for NSP are characterized by a large dispersion, due to optimization challenges: this is especially true when exploiting neural CC modules (Neural-Symbolic, green), but it can also be observed in combination with symbolic CC modules (Symbolic-Symbolic, brown). Conversely, neural NSP, even though often showing unsatisfactory performance, is characterized by a much smaller inter-experiment variance. Table 1 summarizes performance for each task. Overall, the Symbolic-Symbolic category dominates over other groups, with the exception of task 5, where it performs slightly worse than the Symbolic-Neural family. When observing constraint accuracy alone, a downstream Symbolic NSP module of-

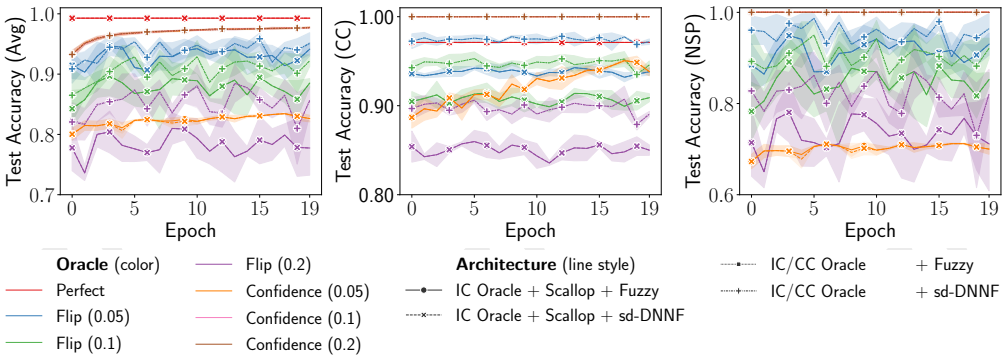

Figure 3: **Q3**. Accuracies for *Task 4* with oracular predictors. Oracle types (perfect, flip, confidence) are described in Section 6.

ten negatively affects performance, with task 4 presenting the highest delta between Symbolic-Symbolic and Symbolic-Neural. Conversely, when focusing on next state prediction, an upstream Symbolic CC module is beneficial to a downstream Symbolic temporal reasoning module, however this clear-cut performance improvement is flipped when the NSP module is Neural. This behavior hints at complex interactions between CC and NSP modules, where architectural choices bi-directionally affect both components. Other hyper-parameters have negligible effect. Early stopping is almost always triggered near the end of training, hinting at a possibly incomplete convergence. Temperature calibration is often selected, but performance improvements are minor (cfr. with Fig. 2), and the semantic loss proposed by [Umili *et al.*, 2023] is not significantly different than traditional binary cross-entropy in our setting. Variances across three runs for each set of hyper-parameters are low. Comparing this observation to Fig. 2 (which, instead, highlights hyper-parameter-based variance), it can be observed that convergence in multi-stage neuro-symbolic sequence classification is heavily dependent on architectural choices, but relatively unaffected by parameter initialization. Task 3 is an exception and it presents a failure case of the Symbolic-Symbolic family, due to training instability. When able to converge, however, it is the only approach capable of improving over random guessing for sequence classification for this task (we reported this sample result surrounding the model name with brackets).

6.2 The Impact of Oracular Predictors

Fig. 3 shows the effect on an exemplar task (Task 4) of oracular predictors on symbolic temporal reasoning modules. Figures 2, 3 and 4, in the supplementary materials, show the same plots for every task. In general, the trainable temperature parameters are quickly optimized, with curves following a mostly horizontal trend. However, flip oracles (blue, green and purple lines) are characterized both by larger variance and inter-epoch oscillations, compared to confidence oracles (orange, pink and brown), which present a remarkably stable behavior. This effect is present in every task, regardless of reasoning difficulty. Noise injection, however, affects the two oracles differently: performance for flip oracles degrades linearly when increasing noise, while confidence oracles are characterized by non-linear behavior. Small amounts of noise do not affect confidence oracles appreciably, while larger amounts tend to harm performance more than the flip oracle. This effect is more evident for harder reasoning tasks: for instance, in task 4, a noise of $p = 0.1$ is enough to cause random guessing performance (Fig. 5 in the Appendix). With oracles, the effect of different automata encodings is virtually non-existent (fuzzy and sd-DNNF markers overlap almost everywhere). IC oracles are consistently more affected by noise, compared to IC/CC ones. This behavior seems counterintuitive, as the Scallop module performs near-exact inference; however, it is in line with the hypothesis of uncertainty accumulation over multiple steps of reasoning. When focusing on constraint performance (Fig. 3 in the supplementary materials), flip oracles tend to have stable performance, while confidence oracles can exploit temperature parameters to achieve a small learning capacity (slight upward trend across epochs). In general, flip oracles are more robust to noise, with confidence oracles achieving unsatisfactory performance (outside plot boundaries, see Fig. 5 in the Appendix) for harder tasks or higher ($p > 0.05$) degrees of uncertainty. This behavior clashes with the desirable property of predictive confidence correlating with uncertainty, which is one of the advantages of neuro-symbolic AI, compared to (uncalibrated) neural networks. Flip oracles achieving better performance than confidence oracles, for similar levels of noise, hint at the fact that an overconfidently-wrong classifier can be more successful than a reluctantly-correct one, in temporal reasoning settings.

7 Conclusions

We extended knowledge-driven sequence classification to a relational setting, introducing a novel benchmarking framework and baselines for neural-only and multi-stage neuro-symbolic methods in a knowledge-driven, and long-temporal-horizon regime. Experiments outline a challenging setting, where neural networks struggle to generalize and state-of-the-art neuro-symbolic methods fall short when stacked: temporal reasoners extended to a relational domain can fail even with full background knowledge available, and general-purpose first-order reasoners suffer from training instabilities in recurrent settings. We argue that the proposed benchmarking framework can benefit the neuro-symbolic and temporal reasoning communities, further pushing research boundaries towards more expressive frameworks, tighter neuro-symbolic integration, and more robust time-driven approaches.

Acknowledgements

M.L. was supported by CAI4DSA actions (Collaborative Explainable neuro-symbolic AI for Decision Support Assistant), PARTENARIATO ESTESO "Future Artificial Intelligence Research - FAIR", SPOKE 1 "Human-Centered AI" Università di Pisa, CUP B13C23005640006. The scholarship by L.S.L. was funded by the Italian Ministry of University and Research (DM 351/2022, PNRR). S.M. was supported by the University of Siena (Piano per lo Sviluppo della Ricerca - PSR 2024, F-NEW FRONTIERS 2024), under the project "TIme-driveN StatEful Lifelong Learning" (TINSELL) and also by the project "CONSTR: a COllectionless-based Neuro-Symbolic Theory for learning and Reasoning", PARTENARIATO ESTESO "Future Artificial Intelligence Research - FAIR", SPOKE 1 "Human-Centered AI" Università di Pisa, "NextGenerationEU", CUP I53C22001380006. L.S.L. scholarship was funded by the Italian Ministry of University and Research (DM 351/2022, PNRR). We kindly thank Luc De Raedt and Nikolaos Manginas for their insightful comments.

References

[Angluin, 1982] Dana Angluin. Inference of reversible languages. *Journal of the ACM (JACM)*, 29(3):741–765, 1982.

[Badreddine *et al.*, 2022] Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artificial Intelligence*, 303:103649, 2022.

[Besold *et al.*, 2021] Tarek R Besold, Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luis C Lamb, Priscila Machado Vieira Lima, Leo de Penning, et al. Neural-symbolic learning and reasoning: A survey and interpretation 1. In *Neuro-Symbolic Artificial Intelligence: The State of the Art*, pages 1–51. IOS press, 2021.

[Camacho and McIlraith, 2019] Alberto Camacho and Sheila A McIlraith. Learning interpretable models expressed in linear temporal logic. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 621–630, 2019.

[Chakraborty *et al.*, 2022] Indrasis Chakraborty, Brian M Kelley, and Brian Gallagher. Device classification for industrial control systems using predicted traffic features. *Frontiers in Computer Science*, 4:777089, 2022.

[Chavira and Darwiche, 2008] Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6-7):772–799, 2008.

[Darwiche and Marquis, 2002] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[De Giacomo and Vardi, 2013] Giuseppe De Giacomo and Moshe Y Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Ijcai*, volume 13, pages 854–860, 2013.

[De Raedt *et al.*, 2016] Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis lectures on artificial intelligence and machine learning*, 10(2):1–189, 2016.

[D'Antoni and Veanes, 2017] Loris D'Antoni and Margus Veanes. The power of symbolic automata and transducers. In *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pages 47–67. Springer, 2017.

[Galassi and Giordana, 2005] Ugo Galassi and Attilio Giordana. Learning regular expressions from noisy sequences. In *International Symposium on Abstraction, Reformulation, and Approximation*, pages 92–106. Springer, 2005.

[Gnecco *et al.*, 2015] Giorgio Gnecco, Marco Gori, Stefano Melacci, and Marcello Sanguineti. Foundations of support constraint machines. *Neural computation*, 27(2):388–480, 2015.

[Green *et al.*, 2007] Todd J Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 31–40, 2007.

[Ivaturi *et al.*, 2021] Praharsh Ivaturi, Matteo Gadaleta, Amitabh C Pandey, Michael Pazzani, Steven R Steinhubl, and Giorgio Quer. A comprehensive explanation framework for biomedical time series classification. *IEEE journal of biomedical and health informatics*, 25(7):2398–2408, 2021.

[Kimmig *et al.*, 2017] Angelika Kimmig, Guy Van den Broeck, and Luc De Raedt. Algebraic model counting. *Journal of Applied Logic*, 22:46–62, 2017.

[LeCun, 1998] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[Li *et al.*, 2023] Ziyang Li, Jiani Huang, and Mayur Naik. Scallop: A language for neurosymbolic programming. *Proceedings of the ACM on Programming Languages*, 7(PLDI):1463–1487, 2023.

[Luo *et al.*, 2022] Weilin Luo, Hai Wan, Delong Zhang, Jianfeng Du, and Hengdi Su. Checking ltl satisfiability via end-to-end learning. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–13, 2022.

[Luo *et al.*, 2024] Weilin Luo, Pingjia Liang, Junming Qiu, Polong Chen, Hai Wan, Jianfeng Du, and Weiyuan Fang. Learning to check ltl satisfiability and to generate traces via differentiable trace checking. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 996–1008, 2024.

[Maene *et al.*, 2024] Jaron Maene, Vincent Derkinderen, and Luc De Raedt. On the hardness of probabilistic neurosymbolic learning. *arXiv preprint arXiv:2406.04472*, 2024.

[Manginas *et al.*, 2024] Nikolaos Manginas, George Paliouras, and Luc De Raedt. Nesya: Neurosymbolic automata. *arXiv preprint arXiv:2412.07331*, 2024.

[Manhaeve *et al.*, 2018] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc

De Raedt. Deepproblog: Neural probabilistic logic programming. *Advances in neural information processing systems*, 31, 2018.

[Manhaeve *et al.*, 2021] Robin Manhaeve, Giuseppe Marra, and Luc De Raedt. Approximate inference for neural probabilistic logic programming. In *KR*, pages 475–486, 2021.

[Marra *et al.*, 2024] Giuseppe Marra, Sebastijan Dumančić, Robin Manhaeve, and Luc De Raedt. From statistical relational to neurosymbolic artificial intelligence: A survey. *Artificial Intelligence*, page 104062, 2024.

[Mukherjee *et al.*, 2022] Prasita Mukherjee, Haoteng Yin, Susheel Suresh, and Tiark Rompf. Octal: Graph representation learning for ltl model checking. *arXiv preprint arXiv:2207.11649*, 2022.

[Nethercote *et al.*, 2007] Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack. Minizinc: Towards a standard cp modelling language. In *International Conference on Principles and Practice of Constraint Programming*, pages 529–543. Springer, 2007.

[Netzer *et al.*, 2011] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 4. Granada, 2011.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. In *18th annual symposium on foundations of computer science (sfcs 1977)*, pages 46–57. ieee, 1977.

[Roesener *et al.*, 2016] Christian Roesener, Felix Fahrenkrog, Axel Uhlig, and Lutz Eckstein. A scenario-based assessment approach for automated driving by using time series classification of human-driving behaviour. In *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*, pages 1360–1365. IEEE, 2016.

[Rozier and Vardi, 2007] Kristin Y Rozier and Moshe Y Vardi. Ltl satisfiability checking. In *International SPIN Workshop on Model Checking of Software*, pages 149–167. Springer, 2007.

[Umili and Capobianco, 2024] Elena Umili and Roberto Capobianco. Deepdfa: Automata learning through neural probabilistic relaxations. In *ECAI 2024*, pages 1051–1058. Ios Press, 2024.

[Umili *et al.*, 2023] Elena Umili, Roberto Capobianco, and Giuseppe De Giacomo. Grounding ltlf specifications in image sequences. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 19, pages 668–678, 2023.

[van Krieken *et al.*, 2022] Emile van Krieken, Erman Acar, and Frank van Harmelen. Analyzing differentiable fuzzy logic operators. *Artificial Intelligence*, 302:103602, 2022.

[van Krieken *et al.*, 2024] Emile van Krieken, Pasquale Minervini, Edoardo M Ponti, and Antonio Vergari. On the independence assumption in neurosymbolic learning. *arXiv preprint arXiv:2404.08458*, 2024.

[Veanes *et al.*, 2010] Margus Veanes, Nikolaj Bjørner, and Leonardo De Moura. Symbolic automata constraint solving. In *Logic for Programming, Artificial Intelligence, and Reasoning: 17th International Conference, LPAR-17, Yogyakarta, Indonesia, October 10-15, 2010. Proceedings 17*, pages 640–654. Springer, 2010.

[Veanes, 2013] Margus Veanes. Applications of symbolic finite automata. In *Implementation and Application of Automata: 18th International Conference, CIAA 2013, Halifax, NS, Canada, July 16-19, 2013. Proceedings 18*, pages 16–23. Springer, 2013.

[Walke *et al.*, 2021] Homer Walke, Daniel Ritter, Carl Trimbach, and Michael Littman. Learning finite linear temporal logic specifications with a specialized neural operator. *arXiv preprint arXiv:2111.04147*, 2021.

[Wang *et al.*, 2024] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[Xiao *et al.*, 2017] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[Xie *et al.*, 2021] Yaqi Xie, Fan Zhou, and Harold Soh. Embedding symbolic temporal knowledge into deep sequential models. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4267–4273. IEEE, 2021.