Deep Weighted Finite Automata

Francesco Casone, Rafael Peñaloza

University of Milano-Bicocca, Milan, Italy

Abstract

Temporal logics and their automata-based counterparts have risen as languages for reasoning about processes and verifying their properties. Increasingly, though, the events in a process refer to observations perceived and classified by a neural network, calling for a mixed formalism capable of dealing with the uncertainty of these classifications. We propose Deep Weighted Finite Automata, a new neuro-symbolic architecture combining formal temporal specifications and neural classifications. We also show how to make probabilistic inferences and compute the most likely execution given a sequence of perceptions.

Keywords

neuro-symbolic AI, weighted automata, temporal reasoning

1. Introduction

The integration of neural perception with symbolic reasoning has emerged as one of the central challenges in modern Artificial Intelligence. While deep learning models excel at processing raw, continuous data, they inherently lack formal guarantees and explicit reasoning capabilities. Conversely, symbolic systems provide rigorous, interpretable logical structures, yet are constrained to operate over discrete symbolic domains. Neurosymbolic AI (NeSy AI) seeks to bridge this gap by combining the complementary strengths of both paradigms [1]. A particularly compelling setting for such integration is that of temporal and sequential data, where the system must not only perceive individual observations but reason coherently about their evolution over time. Domains such as robotics, business process monitoring (BPM), and autonomous systems require models capable of verifying whether a sequence of perceived events complies with formally specified behavioral constraints. Linear Temporal Logic over Finite Traces (LTL_f) [2] provides a well-founded formalism for expressing such constraints, and its automata-theoretic correspondence makes it a natural candidate for integration within computational architectures operating on sequential data.

Several recent works have explored the combination of neural perception with temporal symbolic reasoning, each making distinct design choices that shape their expressive power, their handling of uncertainty, and the range of inference tasks they support. Umili et al. proposed a a modular neural-DFA architecture to ground LTL_f specifications in image sequences. In this model, a neural network evaluates, for each observation, the degree to which atomic propositions hold, producing fuzzy truth values that drive the transitions of a fixed DFA compiled from the given LTL_f formula; algebraic fuzzy operators are used to ensure end-to-end differentiability. Although this approach successfully couples neural perception with a deterministic logical structure, the adoption of fuzzy semantics provides no formal probabilistic interpretation of the model's output, and the absence of a principled stochastic framework precludes the natural application of well-established inference techniques such as MAP estimation or likelihood-based reasoning. NeSyA [4] takes a complementary direction, integrating neural perception with Symbolic Finite Automata interpreted as non-stationary Markov chains, and performing probabilistic inference via weighted model counting. This formulation enables explainable, differentiable reasoning about whether a trace satisfies a given specification. However, NeSyA grounds its constraints

Joint Workshop on Statistics and Knowledge Integration for Logic, Learning, Ethical Decisions, and LLMs (SKILLED-LLMs 2026), co-located with FLoC 2026, Lisbon, Portugal

francesco.casone@mail.polimi.it (F. Casone); rafael.penaloza@unimib.it (R. Peñaloza)

https://rpenalozan.github.io/ (R. Peñaloza)

0009-0001-7651-9410 (F. Casone); 0000-0002-2693-5790 (R. Peñaloza)

in propositional logic rather than directly encoding LTL_f temporal specifications, operating therefore with a temporal relaxation of propositional logic rather than a native temporal formalism.

In a related line of work, DeepDFA [5] and Visual Reward Machines [6, 7] explore learnable NeSy temporal architectures. In particular, DeepDFA pursues the task of learning the probabilistic relaxation of a parametric DFA from data via differentiable operations, introducing probabilistic reasoning on symbolic structures.

Existing temporal neurosymbolic approaches present a fragmented landscape: fuzzy temporal architectures lack formal probabilistic semantics and restrict the range of applicable inference techniques; probabilistic temporal systems either conflate observational and dynamic uncertainty or do not natively encode LTL_f constraints. In this work, we address these limitations by proposing *Deep Weighted Finite Automata (DWFA)*, a modular neurosymbolic architecture that combines neural perception with symbolic temporal reasoning. The key design insight is the dynamic assignment of automaton transition weights at runtime, driven by the confidence values produced by the neural component. This mechanism bridges the two modules while fully preserving the determinism of the underlying DFA structure derived from LTL_f specifications, ensuring that uncertainty in the model arises exclusively from perception and not from the logical dynamics and shifting the computational logical structure to a Weighted Finite Automaton, allowing quantitative reasoning to work in parallel with symbolical reasoning. Also, by constructing the WFA over an adaptable semiring, a single unified architecture natively supports multiple qualitatively different inference procedures without any structural modification.

In summary, the main contribution of this paper is the definition of DWFA, a modular neurosymbolic architecture for temporally related sequences of data with the following key properties:

1. a WFA encoding the logical structure of the system, performing quantitative and logical reasoning and providing flexibility in inference;
2. a dynamic weight assignment mechanism integrating observational uncertainty in the model and yet retaining the determinism of the system dynamics under certain observations;
3. the assumption of a probabilistic interpretation of perceptual uncertainty employed to reason in an interpretable and efficient manner.

The remainder of the paper is organized as follows: in Section 2 we revise some preliminary concepts on Weighted Finite Automata and Linear Temporal Logic over Finite Traces, in Section 3 we present the framework of Deep Weighted Finite Automata, in Section 4 we show how it is used for different reasoning tasks and Section 5 presents a complete numerical example of an instance of the model, then finally in Section 6 we conclude and discuss directions for future work.

2. Preliminaries

We briefly introduce the main notions of weighted finite automata and temporal logic on finite traces needed to understand the rest of the paper.

2.1. Weighted Finite Automata

Finite automata [8] provide a standard formalism to model sequential processes by evolving through a finite set of states in response to an input sequence. In their classical form, they define a Boolean notion of acceptance: a sequence is either accepted or rejected depending on the final state reached by the computation. While this formulation is sufficient for reasoning over symbolic sequences, it is not expressive enough to capture uncertainty or graded information, which naturally arise when dealing with data-driven systems. We hence consider Weighted Finite Automata (WFA) [9], which extend classical automata by associating quantitative values to transitions and states. This allows the model to assign a numerical value to each input sequence rather than a binary decision.

WFAs are defined over a *semiring*, which is an algebra $(R, \oplus, \otimes, \bar{0}, \bar{1})$ composed of a set R equipped with two associative binary operations $\oplus$ (addition) and $\otimes$ (multiplication), with distinct neutral elements $\bar{0}$ and $\bar{1}$, where $\oplus$ is commutative, $\otimes$ distributes over $\oplus$, and $\bar{0}$ annihilates over $\otimes$. Given such

a semiring, a WFA is a tuple $A = (Q, \Sigma, \delta, I, F)$ where Q is a finite set of *states*; Σ is a finite *alphabet*; $\delta : Q \times \Sigma \times Q \to R$ is the *weighted transition function*; $I : Q \to R$ is an *initial weight function*; and $F : Q \to R$ is a *final weight function*.

Let $w = a_1, a_2, \ldots, a_T \in \Sigma^*$. Each possible *path* $\pi = q_0, q_1, \ldots, q_T \in Q^{T+1}$ for w defines a path weight

$$\mathsf{wt}(\pi) = I(q_0) \otimes \left(\bigotimes_{t=1}^{T} \delta(q_{t-1}, a_t, q_t) \right) \otimes F(q_T).$$

The *behaviour* of A over w is then

$$A(w) = \bigoplus_{\pi \in Paths(w)} \mathsf{wt}(\pi).$$

A thus defines a function $[[A]] : \Sigma^* \to R$. The computation of this function clearly depends on the chosen semiring; therefore, different semirings correspond to different interpretations, resulting in distinct weights assigned to a given input sequence. A WFA is *deterministic* if the finite automaton obtained through the characteristic function of δ is deterministic.

2.2. LTL and LTL$_f$

Linear Temporal Logic [10] is a temporal extension of propositional logic used to describe properties that evolve over time. LTL$_f$ [2] is a variant of this logic that considers only finite traces; i.e., where every execution eventually stops. Given a set $\mathcal{P}$ of atomic propositions, the class of LTL$_f$ formulas ϕ over $\mathcal{P}$ is defined through the grammar: $\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid X\phi \mid \phi_1 U \phi_2, \mid \top$, where $p \in \mathcal{P}$.

The propositional operators $\wedge$ and $\neg$ and the temporal operators X (Next) and U (Until) form a complete basis to define all the classical propositional operators and other temporal operators, such as, N (Weak Next) $N\phi \equiv \neg X \neg \phi$, R (Release) $\phi_1 R \phi_2 \equiv \neg(\neg\phi_1 U \neg\phi_2)$, G (Globally) $G\phi \equiv \bot R \phi$ and F (Eventually) $F\phi \equiv \top U \phi$.

A *valuation* is a subset of $\mathcal{P}$ expressing which variables are considered true. A *trace* is a sequence $\rho = \rho[0], \rho[1], \ldots, \rho[T]$ of valuations; its *length* is $|\rho| = T + 1 < \infty$. The *satisfiability* of a formula ϕ at time i in trace ρ (denoted $\rho, i \models \phi$) is defined recursively on the structure of ϕ as follows: (i) $\rho, i \models q \in \mathcal{P}$ iff $q \in \rho[i]$; (ii) $\rho, i \models \neg\phi$ iff $\rho, i \not\models \phi$; (iii) $\rho, i \models \phi_1 \wedge \phi_2$ iff $\rho, i \models \phi_1$ and $\rho, i \models \phi_2$; (iv) $\rho, i \models X\phi$ iff $\rho, i + 1 \models \phi$; and (v) $\rho, i \models \phi_1 \, U \, \phi_2$ iff $\exists j \geq i$ s.t. $\rho, j \models \phi_2$ and $\forall k, i \leq k < j, \rho, k \models \phi_1$.

It is well known that for every LTL$_f$ formula ϕ it is possible to construct a finite automaton which accepts exactly the traces that satisfy ϕ [11]. The size of the automaton A_ϕ, in the worst case, might be double exponential in ϕ [2]. In practice, however, A_ϕ is often moderate in dimension and the double exponential relation between A_ϕ and ϕ is generally not triggered, resulting in feasible computations. In addition, scalable techniques have been developed for this task [12, 13, 14]. Our goal is to use this characterisation to reason with LTL$_f$ specifications associated to neural symbol perception.

3. Deep Weighted Finite Automata

We propose a novel framework that combines the potential of neural perception with the coherence of symbolic temporal reasoning. In particular, we propose Deep Weighted Finite Automata, a modular architecture composed of

1. a *neural component* that performs perception on subsymbolic data, carrying out a multiclass classification task to map data to the discrete symbolic domain of reference. The definition of the neural network is contingent upon the specific use case. In particular, the neural architecture acts on a single observation at a time, not on traces, and can hence be chosen to better fit the data under consideration; and

2. a recurrent *logic component* consisting of a WFA with probabilistic semantics, whose weights are assigned at runtime by the neural component. At each time step, the weight of a transition of the automaton caused by the observation of a symbol *s* will be the confidence value in the neural perception of *s*. The structure of the automaton, instead, is built from an LTL_f specification.

The choice of a WFA with dynamic weights provides a higher degree of generalization on the reasoning tasks and allows observational uncertainty to be integrated without losing the determinism of the logical structure; therefore without integrating any new dynamical uncertainty. Using WFAs facilitates a wider range of operations by a choice of the semiring on which the WFA is constructed.

Deep Weighted Finite Automata work under the *assumption* that the neural algorithm performing perception produces a stochastic vector over the possible symbolic interpretations of the given input. A *stochastic vector* is a vector of non-negative entries which add to 1 [15]. Thus, the confidence values of the neural network can be seen as the probability of the input data point being mapped to each symbol in the alphabet of the logical module. While it allows the system to work under a probabilistic interpretation, this assumption is not very restrictive on the framework. The vast majority of real-world neural networks designed for multi-class classification tasks, produce a stochastic vector as their final output through a Softmax activation function [16]. Moreover, if the properties of stochastic vectors are not respected, it is always possible to map a non-stochastic vector to a stochastic one through L1-normalization if the non-stochastic vector only has non-negative entries. The assumption is therefore reduced to having a neural component producing non-negative vectors. We note that many efforts are made to obtain *calibrated* multi-classifiers, which respect this assumption as well [17].

Formally a Deep Weighted Finite Automaton is a tuple $DWFA = (Q, \Sigma, \delta, \mathbf{w_I}, \mathbf{w_F}, f_\theta)$ where:

- Q is a finite set of size n of *logical states* in an arbitrary but fixed order;
- Σ is a finite *alphabet* of propositional symbols;
- $\delta : Q \times \Sigma \to Q$ is the *deterministic transition function*;
- $\mathbf{w_I} \in \mathbb{R}^n$ is the *initial state vector*, it is a stochastic vector;
- $\mathbf{w_F} \in \mathbb{B}^n$ is the final *vector* of *accepting states*, a binary vector; and
- f_θ the neural network mapping subsymbolic data to the symbolic domain.

The binary vector $\mathbf{w_F}$ defines a set F of *accepting states* corresponding to the positions in $\mathbf{w_F}$ having the value 1.

The automaton preserves a *state vector*, which is a stochastic vector, throughout its execution. This vector is initialised with $\mathbf{w_I}$ and updated through the perceptions of f_θ and the transition function δ as explained next.

Recall that in deterministic (classical) automata, the transition function specifies, given a state and a symbol, the new state into which the automaton is located. We preserve the same intuition, except that we have no certainty about the current state (just a probability distribution), nor about the symbol being read (given the uncertainty of the classifier). To keep track of the updated probabilities at each step of the automaton, we see the transition function as a class of *transition matrices* $T_a \in \mathbb{B}^{n \times n}$; one for each $a \in \Sigma$. Since δ is a *deterministic* transition function, each row in these matrices has exactly one non-zero entry.

Consider now a finite sequence $o_1, \ldots, o_T$ of input objects (to be read by f_θ), and define, for each $t, 0 < t < T$, the quantities

$$\begin{aligned} \mathbf{p}_t(a) &= f_\theta(o_t)_{|a}, \\ T(\mathbf{p}_t) &= \bigoplus_{a \in \Sigma} \mathbf{p}_t(a) T_a. \end{aligned}$$

In words, $\mathbf{p}_t(a)$ is the probability assigned by the classifier that the input at timepoint t is in fact the symbol a; we will also denote this as $p(a \mid o_t)$ when it is important to know the input object. Given these values, $T(\mathbf{p}_t)$ represents the weighted transition matrix at time t, computed as a combination of the deterministic transition matrices for each symbol weighted by the probability of observing that symbol.

Thus, the dependence of the transition matrices is shifted from symbols $a \in \Sigma$ to neural observations $\mathbf{p}_t$. This definition of the weighted transition matrices allows their dynamic computation at runtime; that is, it allows to compute them at each time conditioned on the corresponding sub-symbolic observation.

Consider for now a generic certain observation o, which is classified as the symbol a_i with probability 1; i.e., the classifier is certain about its observation. We can see $\mathbf{p}_t$ as a vector

$$\mathbf{p}_t = \big[p(a_1 \mid o_t), \ldots, p(a_i \mid o_t), \ldots, p(a_m \mid o_t)\big] = \big[0, \ldots, 1, \ldots, 0\big]$$

where $m = |\Sigma|$. Restricting to a semiring $([0,1], \bigoplus, \bigotimes, 0, \bar{1})$ yields

$$T(\mathbf{p}_t) = \bigoplus_{a \in \Sigma} \mathbf{p}_t(a) T_a = \bigoplus_{a \in \Sigma} T_a \delta_{a,a_i} = T_{a_i}$$

In other words, independently of the chosen semiring addition $\oplus$ chosen, the weighted transition matrices collapse to the deterministic transition matrix of the symbol observed with certainty, thus cancelling all contributions from symbols that are certainly not observed. This is as expected, showing that DWFA are a generalisation of deterministic finite automata, when no uncertainty is present. In the following, we consider only semirings where the addition neutral element is 0.

4. Reasoning

We now consider two important reasoning tasks which can be solved within this formalism. The first refers to satisfying a given specification, while the second considers the problem of updating the probabilities of an observation given further evidence.

4.1. Trace Verification and Monitoring

Problem Formulation We first consider the task of classifying a sequence of observations as compliant or not with a certain LTL$_f$ specification. Being the input subsymbolic trace neurally perceived by the Deep Weighted Finite Automaton, each possible symbolic trace is associated to the subsymbolic one with a definite probability, hence, the compliance of the trace with the temporal rules will be computed as a probability.

Inference Procedure Consider a DWFA $\mathcal{D} = (Q, \Sigma, \delta, \mathbf{w_I}, \mathbf{w_F}, f_\theta)$ over the semiring $([0,1], +, \times, 0, 1)$, which is a variant of the probability semiring [18] where the domain is restricted to the unit interval to preserve the probabilistic interpretation of the values throughout the inference procedure. It is straightforward that the multiplication is closed in the interval. The unit interval is, in general, not closed under the addition. Yet, as we will see, limiting the operations to stochastic vectors and a deterministic transition function guarantees that all calculations remain in this interval. In particular, note that every output of the classifier f_θ is a probability distribution over the available symbols Σ.

Let φ be an LTL$_f$. Recall that a trace of sub-symbolic observations $\rho = o_1, ..., o_T$ produces neural observations $\mathbf{p}_1, \ldots, \mathbf{p}_T$ where each $\mathbf{p}_i = f_\theta(o_i)$. Using the notation T_a to indicate the deterministic transition matrices of the automaton, one define the weighted transition matrix at each time step as $T(\mathbf{p}_t) = \sum_{a \in \Sigma} \mathbf{p}_t(a) T_a$, which defines the state vector update operation $\mathbf{s}_{t+1} = \mathbf{s}_t T(\mathbf{p}_t)$.

By applying this update iteratively through the execution of the automaton, we obtain the *final state vector*

$$\mathbf{s}_T = \mathbf{w_I} \bigotimes_{t=1}^{T} T(\mathbf{p}_t) = \mathbf{w_I} \prod_{t=1}^{T} T(\mathbf{p}_t) = \mathbf{w_I} \prod_{t=1}^{T} \left(\sum_{a \in \Sigma} \mathbf{p}_t(a) T_a \right).$$

This final state vector $\mathbf{s}_T$ represents the degree of belief of the system being in each logical state at the last time step of the trace; in particular, the i-th component $s_T[i]$ of the vector quantifies the probability that the system is in the state q_i at time T. Consequently, the probability $P(\rho \models \varphi)$ of a trace being

accepted by the automaton and, therefore, being compliant with the temporal specification, is the addition over the probabilities of the system ending in a final accepting state, that is, via scalar product (on $\mathbb{R}^{|\Sigma|}$) between $\mathbf{s}_T$ and the vector of accepting states $\mathbf{w_F}$ exploiting the definition of addition over the semiring of reference. That is,

$$P(\rho \models \varphi) = \mathbf{s_T} \cdot \mathbf{w_F} = \bigoplus_{q_i \in F} \mathbf{s_T}[i] = \sum_{q_i \in F} \mathbf{s_T}[i].$$

The complete inference procedure is therefore described by the formula:

$$P(\rho \models \varphi) = \mathbf{w_I} \left(\bigotimes_{t=1}^{T} \bigoplus_{a \in \Sigma} p_t(a) T_a \right) \cdot \mathbf{w_F} = \mathbf{w_I} \left(\prod_{t=1}^{T} \sum_{a \in \Sigma} p_t(a) T_a \right) \cdot \mathbf{w_F}$$

Importantly, this calculation is based on pure matrix multiplication. Hence it is linear and differentiable and can be exploited to train the classifier f_θ via gradient descent. In particular, considering a fixed logical structure, it is possible to train the neural component of DWFA in two settings:

(i) *strongly supervised* setting, the model is trained on sequences of data presenting labels for every element in the input sequence. The symbolic assignment of every subsymbolic observation in the input sequence is given, the training process is therefore supervised at each time step;

(ii) *weakly supervised* setting, the dataset presents a single label for each sequence of data, specifying whether the full trace is compliant or not with the temporal specification taken into consideration. The training process relies on the classification of complete sequences of data rather than on the symbolic classification of single observations.

In a business application, one typically has several known conforming traces, for which the weakly supervised setting is inherently adequate. Clearly, the properties of the classifier learned this way cannot fully guarantee what an *ad-hoc* classifier may produce, but the implicit learning leads to a model better targetted to the specific application.

As mentioned already, the state vector of the DWFA is guaranteed to be a stochastic vector throughout the entire process, making the model inherently *end-to-end interpretable*. The component $\mathbf{s}_t[i]$ of the state vector represents, indeed, the confidence of the model being in the logical state q_i at time t.

Online Monitoring Consider now a setting in which the model is presented with partial traces, and the task is to quantify the probability that the partial trace satisfies the temporal specification φ of the system. Due to the end-to-end interpretability of the model, the inference procedure is not modified. That is, given a subsymbolic input $\rho = o_0, o_1, \ldots, o_T$, we denote by $\rho_{t' \leq T} = o_0, \ldots, o_{t'}$ the partial trace considering the first t' time steps with $t' \leq T$ and compute:

$$P(\rho_{t' \leq T} \models \varphi) = \mathbf{s}_{t'} \cdot \mathbf{w_F} = \mathbf{w_I} \left(\bigotimes_{t=1}^{t'} \bigoplus_{a \in \Sigma} \mathbf{p}_t(a) T_a \right) \cdot \mathbf{w_F} = \mathbf{w_I} \left(\prod_{t=1}^{t'} \sum_{a \in \Sigma} \mathbf{p}_t(a) T_a \right) \cdot \mathbf{w_F}$$

What this computation tells us is, if the execution ended at this transition, what is the probability of satisfying the specification? This probability evolves during the execution. This can be further refined to the probabilities of permanently satisfying or violating the property by substituting $\mathbf{w_F}$ with the vector of states which belong to accepting or rejecting cliques [19, 20, 21].

4.2. MAP Symbolic Trace

Problem Formulation We now consider a complementary inference task: given a subsymbolic input sequence, we seek the most probable symbolic trace that is consistent with both the neural observations and the LTL$_f$ specification.

Formally, given a DWFA $\mathcal{D} = (Q, \Sigma, \delta, \mathbf{w}_I, \mathbf{w}_F, f_\theta)$ encoding a temporal specification φ and a subsymbolic trace $\rho = o_0, o_1, \ldots, o_T$, the goal is to compute:

$$a^*_{0:T} = \arg\max_{a_{0:T} \in \Sigma^T} P(a_{0:T} \mid \rho, \varphi)$$

where $a^*_{0:T} = a^*_0, a^*_1, \ldots, a^*_T$ is the optimal symbolic trace. This constitutes a constrained *maximum a posteriori* (MAP) estimation problem under the temporal specification φ. In words, since the classifier f_θ is uncertain of all its observations, given a sequence of objects, we are not sure which symbols these observations correspond to. However, the specification disallows some possibilities, while other observations may influence the likelihood of a given symbol at a given timepoint. Our goal is to find the most likely sequence of symbols that corresponds to the observations, assuming that the trace satisfies the specification.

By Bayes' theorem, the posterior factorizes as:

$$P(a_{0:T} \mid \rho, \varphi) \propto P(\rho \mid a_{0:T}, \varphi)\, P(a_{0:T} \mid \varphi), \quad P(a_{0:T}|\varphi) = \begin{cases} c & \text{if } a_{0:T} \models \varphi \\ 0 & \text{otherwise} \end{cases}, \quad c \in \mathbb{R}$$

The prior $P(a_{0:T} \mid \varphi)$ encodes the structural hard constraint imposed by the automaton: it is uniform over compliant traces and zero otherwise. Under this uniform prior and assuming independence across time steps, MAP estimation reduces to Maximum Likelihood Estimation over the language of the DWFA:

$$a^*_{0:T} = \arg\max_{a_{0:T}} P(\rho|a_{0:T}, \varphi) = \arg\max_{a_{0:T} \in L(DWFA)} P(a_{0:T}|\rho) = \arg\max_{a_{0:T} \in L(DWFA)} \prod_{t'=0}^{T} \mathbf{p}_{t'}(a_t)$$

Inference Procedure To solve this problem, the DWFA is instantiated over the Viterbi semiring $([0, 1], \max, \times, 0, 1)$. The inference proceeds in two phases.

Forward pass. The state vector is initialized as $\mathbf{s}_0 = \mathbf{w}_I$ and updated recursively for $t = 0, \ldots, T-1$:

$$\mathbf{s}_{t+1} = \bigoplus_{q \in Q} (\mathbf{s}_t \otimes T(\mathbf{p}_t)) = \max_{q \in Q} \left(\mathbf{s}_t \times \max_{a \in \Sigma} \mathbf{p}_t(a)\, T_a \right)$$

where $T(\mathbf{p}_t) = \bigoplus_{a \in \Sigma} \mathbf{p}_t(a)\, T_a$ is the weighted transition matrix at time t under the Viterbi semiring. Each entry $\mathbf{s}_t[i]$ stores the maximum cumulative weight over all paths of length t ending in state q_i. At each time step t and for each state q, a backpointer $bp_t(q) = (q', a)$ is stored, recording the predecessor state and symbol that realize the maximum:

$$\mathbf{s}_{t+1}[j] = \max_{q_i \in Q} (\mathbf{s}_t[i] \cdot T(\mathbf{p}_t)[i, j])$$

Backtracking. The optimal final state is selected as:

$$q^*_T = \arg\max_{q \in F} \ \mathbf{s}_T \cdot \mathbf{w}_F$$

and the optimal trace is recovered by following the backpointers:

$$(q^*_{t-1},\, a^*_t) = bp_t(q^*_t), \qquad t = T, \ldots, 1$$

yielding $a^*_{0:T} = a^*_0, a^*_1, \ldots, a^*_T$.

During this procedure, by recursively multiplying a sequence of neural probabilities accumulated along a path, values decrease exponentially with the sequence length, making it susceptible to severe numerical underflow in floating-point arithmetic. To ensure numerical stability, the computation can be performed in the logarithmic domain, by defining $\mathbf{l}_t(a) = \log(\mathbf{p}_t(a))$ and $\mathbf{v}_t(a) = \log \mathbf{s}_t(a)$, the semiring operations $(\max, \times)$ transform into $(\max, +)$. This is a typical solution method for Viterbi algorithms [22], of which the proposed inference process can be regarded as a generalization to WFAs.

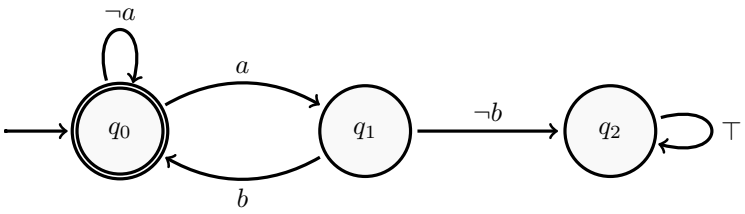

Figure 1: Deterministic finite automaton for the formula "$\varphi = G(a \rightarrow Xb)$"

5. Example

We instantiate a minimal DWFA for the LTL$_f$ formula $\varphi = G(a \rightarrow Xb)$ ("globally if a then next b"), i.e. the trace is accepted iff every observation of a is followed by an observation of b, where $AP = \{a, b\}$ is the set of relevant propositional variables. The DFA corresponding to this formula is depicted in Figure 1.

To ensure a meaningful run of the model, we assume that the system starts with a fixed state q_0, and hence consider the initial state vector $\mathbf{w_I} = \begin{pmatrix}1 & 0 & 0\end{pmatrix}$. Being q_0 the only accepting state, the final vector is $\mathbf{w_F} = \begin{pmatrix}1 & 0 & 0\end{pmatrix}$ and the set of final accepting states is $F = \{q_0\}$. Using these state vectors and a classifier f_θ, the DWFA for the specification φ is defined as $\mathcal{D} = (Q, \Sigma, \delta, \mathbf{w_I}, \mathbf{w_F}, f_\theta)$ where $Q = \{q_0, q_1, q_2\}$; $\Sigma = 2^{AP} = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$; and δ is the transition function from the automaton in Figure 1.

The deterministic transition matrices of the DWFA $\mathcal{D}$ are

$$T_\emptyset = \begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 1\\ 0 & 0 & 1\end{bmatrix} \quad T_a = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0 & 1\end{bmatrix} \quad T_b = \begin{bmatrix}1 & 0 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix} \quad T_{ab} = \begin{bmatrix}0 & 1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}$$

where

$$T_\sigma[i, j] = \begin{cases}1 & \text{if } \delta(q_i, \sigma) = q_j\\ 0 & \text{otherwise}\end{cases}, \quad \text{for } \sigma \in \Sigma$$

We then consider a subsymbolic sequence $\rho = o_1, o_2$. Neural perception produces at each time t a distribution over the four symbols in Σ

$$\mathbf{p}_t = f_\theta(o_t) = \begin{pmatrix}P(\emptyset|o_t) & P(a|o_t) & P(b|o_t) & P(ab|o_t)\end{pmatrix}$$

Concretely, for the sake of the example assume that the neural network f_θ produces the outputs

$$\mathbf{p}_1 = \begin{pmatrix}0.30 & 0.60 & 0.10 & 0.00\end{pmatrix}$$
$$\mathbf{p}_2 = \begin{pmatrix}0.10 & 0.05 & 0.8 & 0.05\end{pmatrix}$$

We now proceed to show the inference procedures in this context, showing numerical calculations at each step.

Trace verification To compute the probability of the subsymbolic trace ρ being compliant with the temporal specification φ, at each time, we will firstly compute the weighted transition matrix and then update the state vector accordingly. Remember that at time $t = 0$, we have the initial state vector $\mathbf{w_I}$. The updates of the state vector proceed as follows.

$t = 1$ We first compute the weighted transition matrix

$$T(\mathbf{p}_1) = \sum_{\sigma \in \Sigma} \mathbf{p}_1(\sigma) T_\sigma = 0.30 T_\emptyset + 0.60 T_a + 0.10 T_b + 0.00 T_{ab}$$

$$= 0.30 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} + 0.60 \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} + 0.10 \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.40 & 0.60 & 0 \\ 0.10 & 0 & 0.90 \\ 0 & 0 & 1 \end{bmatrix}$$

which is used to update the state vector

$$\mathbf{s}_1 = \mathbf{w_I} T(\mathbf{p}_1) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.40 & 0.60 & 0 \\ 0.10 & 0 & 0.90 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.40 & 0.60 & 0 \end{bmatrix}$$

$t = 2$ Following the same steps, we obtain

$$\begin{aligned} T(\mathbf{p}_2) &= \sum_{\sigma \in \Sigma} p_2(\sigma) T_\sigma = 0.10 T_\emptyset + 0.05 T_a + 0.80 T_b + 0.05 T_{ab} \\ &= 0.10 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} + 0.05 \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} + 0.80 \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} + 0.05 \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0.90 & 0.10 & 0 \\ 0.85 & 0 & 0.15 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

$$\mathbf{s}_2 = \mathbf{s}_1 T(\mathbf{p}_2) = \begin{bmatrix} 0.40 & 0.60 & 0 \end{bmatrix} \begin{bmatrix} 0.90 & 0.10 & 0 \\ 0.85 & 0 & 0.15 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.87 & 0.04 & 0.09 \end{bmatrix}$$

Lastly, we compute the acceptance probability of the trace:

$$P(\rho \models \varphi) = \mathbf{s}_2 \cdot \mathbf{w_F} = 0.87 \cdot 1 + 0.04 \cdot 0 + 0.09 \cdot 0 = 0.87.$$

This means that the DWFA $\mathcal{D}$ assigns a probability of 0.87 to the trace ρ being compliant with $\varphi = G(a \rightarrow Xb)$ under the produced neural outputs.

MAP trace calculation We now focus on the computation of the MAP symbolic trace $\sigma^*_{1:2} = (\sigma^*_1, \sigma^*_2)$ subject to satisfying the temporal constraint φ; the following paragraph will show the step by step calculation.

At each step, we define the weighted transition matrix, update the state vector and save the backpointers, and then perform backtracking to recover the MAP symbolic trace $\sigma^*_{1:2}$ and, simultaneously, the MAP sequence of logical states (q^*_1, q^*_2).

$t = 1$ For the weighted transition matrix computation, the maximum operation acts element by element for each entry of the matrix.

$$\begin{aligned} T(\mathbf{p}_1) &= \max_{\sigma \in \Sigma} \mathbf{p}_1(\sigma) T_\sigma \\ &= \max_{\sigma \in \Sigma} \left(0.30 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, 0.60 \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, 0.10 \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, 0 \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \\ &= \begin{bmatrix} 0.30 & 0.60 & 0 \\ 0.10 & 0 & 0.60 \\ 0 & 0 & 0.60 \end{bmatrix} \end{aligned}$$

Using this, we can update of the state vector and compute the backpointers

$$\mathbf{s}_1 = \max_{q \in Q} \mathbf{w_I} T(\mathbf{p}_1) = \max_{q \in Q} \left(\begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.30 & 0.60 & 0 \\ 0.10 & 0 & 0.60 \\ 0 & 0 & 0.60 \end{bmatrix} \right)$$

$$= \begin{bmatrix}\max(0.30,0,0) & \max(0.60,0,0) & \max(0,0,0)\end{bmatrix} = \begin{bmatrix}0.30 & 0.60 & 0\end{bmatrix}$$
$$bp_1(q_0) = (q_0, \emptyset)$$
$$bp_1(q_1) = (q_0, a)$$
$$bp_1(q_2) = (q_0, \emptyset)$$

$t = 2$ Weighted transition matrix computation:

$$\begin{aligned} T(\mathbf{p}_2) &= \max_{\sigma\in\Sigma} \mathbf{p}_2(\sigma) T_\sigma \\ &= \max_{\sigma\in\Sigma} \left(0.10 \begin{bmatrix}1&0&0\\0&0&1\\0&0&1\end{bmatrix}, 0.05 \begin{bmatrix}0&1&0\\0&0&1\\0&0&1\end{bmatrix}, 0.80 \begin{bmatrix}1&0&0\\1&0&0\\0&0&1\end{bmatrix}, 0.05 \begin{bmatrix}0&1&0\\1&0&0\\0&0&1\end{bmatrix} \right) \\ &= \begin{bmatrix}0.80 & 0.05 & 0\\ 0.80 & 0 & 0.10\\ 0 & 0 & 0.80\end{bmatrix} \end{aligned}$$

Update of the state vector and backpointers:

$$\begin{aligned} \mathbf{s}_2 &= \max_{q\in Q} \mathbf{s}_1 T(\mathbf{p}_2) \\ &= \max_{q\in Q} \left(\begin{bmatrix}0.30 & 0.60 & 0\end{bmatrix} \begin{bmatrix}0.80 & 0.05 & 0\\ 0.80 & 0 & 0.10\\ 0 & 0 & 0.80\end{bmatrix} \right) \\ &= \begin{bmatrix}\max(0.24, 0.48, 0) & \max(0.015, 0, 0) & \max(0, 0.06, 0)\end{bmatrix} = \begin{bmatrix}0.48 & 0.015 & 0.06\end{bmatrix} \end{aligned}$$
$$bp_2(q_0) = (q_1, b);$$
$$bp_2(q_1) = (q_0, a);$$
$$bp_2(q_2) = (q_1, \emptyset)$$

Finally, we select the most probable final state, in this case the only accepting state is q_0, so the calculation is trivial:

$$q^*_{t=2} = \arg\max_{q\in F} (\mathbf{s}_2 \cdot \mathbf{w_F}) = \arg\max_{q\in F} \left(\begin{bmatrix}0.48 & 0.015 & 0.06\end{bmatrix} \cdot \begin{bmatrix}1 & 0 & 0\end{bmatrix} \right) = q_0$$

Backtracking

$$q^*_{t=2} = q_0,\ bp_2(q_0) = (q_1, b) \implies q^*_{t=1} = q_1, \sigma^*_2 = b;$$
$$q^*_{t=1} = q_1,\ bp_1(q_1) = (q_0, a) \implies \sigma^*_1 = a.$$

Therefore, the MAP symbolic trace for this example is $\sigma^*_{1:2} = (a, b)$.

In this case, the MAP symbolic trace corresponds exactly to what the classifier f_θ perceived. However, this is not always the case. Consider a new trace o'_1, o'_2 where the NN produces the outputs

$$\mathbf{p}'_1 = \begin{pmatrix}0.10 & 0.30 & 0.60 & 0.00\end{pmatrix}$$
$$\mathbf{p}'_2 = \begin{pmatrix}0.10 & 0.70 & 0.15 & 0.05\end{pmatrix}$$

We see that the classifier, through the independent readings, reports that the symbols in the trace are "ba." However, this trace does not comply with the specification, meaning that if the trace is complying, the classifier has made some error. Computing the MAP allows to detect which of the possible complying traces (including "ab" and "bb") is the most likely. As before, we proceed step by step and then backtrack to find the symbolic trace.

$t = 1$ The weighted transition matrix computation yields

$$\begin{aligned} T(\mathbf{p}'_1) &= \max_{\sigma\in\Sigma} \mathbf{p}'_1(\sigma) T_\sigma \\ &= \max_{\sigma\in\Sigma} \left(0.10 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, 0.30 \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, 0.60 \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, 0 \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \\ &= \begin{bmatrix} 0.60 & 0.30 & 0 \\ 0.60 & 0 & 0.30 \\ 0 & 0 & 0.60 \end{bmatrix}, \end{aligned}$$

from which we can update of the state vector and compute the backpointers

$$\mathbf{s}_1 = \max_{q\in Q} \mathbf{w_I} T(\mathbf{p}'_1) = \max_{q\in Q} \left(\begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.60 & 0.30 & 0 \\ 0.60 & 0 & 0.30 \\ 0 & 0 & 0.60 \end{bmatrix} \right) = \begin{bmatrix} 0.60 & 0.30 & 0 \end{bmatrix}$$

$$\begin{aligned} bp_1(q_0) &= (q_0, b) \\ bp_1(q_1) &= (q_0, a) \\ bp_1(q_2) &= (q_0, \emptyset) \end{aligned}$$

$t = 2$ Weighted transition matrix computation:

$$T(\mathbf{p}'_2) = \max_{\sigma\in\Sigma} \mathbf{p}'_2(\sigma) T_\sigma = \max_{\sigma\in\Sigma} \left(0.10 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, 0.70 \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, 0.15 \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, 0.05 \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 0.15 & 0.70 & 0 \\ 0.15 & 0 & 0.70 \\ 0 & 0 & 0.70 \end{bmatrix}$$

Update of the state vector and backpointers:

$$\begin{aligned} \mathbf{s}_2 &= \max_{q\in Q} \mathbf{s}_1 T(\mathbf{p}'_2) = \max_{q\in Q} \left(\begin{bmatrix} 0.60 & 0.30 & 0 \end{bmatrix} \begin{bmatrix} 0.15 & 0.70 & 0 \\ 0.15 & 0 & 0.70 \\ 0 & 0 & 0.70 \end{bmatrix} \right) \\ &= \begin{bmatrix} \max(0.09, 0.045, 0) & \max(0.42, 0, 0) & \max(0, 0.21, 0) \end{bmatrix} = \begin{bmatrix} 0.09 & 0.42 & 0.21 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} bp_2(q_0) &= (q_0, b); \\ bp_2(q_1) &= (q_0, a); \\ bp_2(q_2) &= (q_1, a) \end{aligned}$$

The most probable final state is the only accepting state (q_0):

$$q^*_{t=2} = \arg\max_{q\in F} (\mathbf{s}_2 \cdot \mathbf{w_F}) = \arg\max_{q\in F} \left(\begin{bmatrix} 0.09 & 0.42 & 0.21 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \right) = q_0$$

Backtracking

$$\begin{aligned} q^*_{t=2} = q_0,\ bp_2(q_0) = (q_0, b) &\implies q^*_{t=1} = q_0, \sigma^*_2 = b; \\ q^*_{t=1} = q_0,\ bp_1(q_0) = (q_0, b) &\implies \sigma^*_1 = b. \end{aligned}$$

The most likely symbolic trace is hence *bb*. This means that the classifier most likely made an error when reading the second symbol, but this error could be corrected using the information of the other reading and the formal specification.

6. Conclusions

This paper introduced Deep Weighted Finite Automata, a modular neurosymbolic architecture for coherent symbolic temporal reasoning over subsymbolic data sequences. The framework combines a neural perception module, mapping continuous observations to probability distributions over symbolic interpretations, with a symbolic reasoning module consisting of a Weighted Finite Automaton whose structure is derived from LTL_f specifications. The key contributions proposed lie in the design of the logic component and of the mechanism bridging it to the neural perception component. The latter works by dynamic assignment of the transition weights of the WFA at runtime, driven by the neural component's confidence values. This mechanism integrates perceptual uncertainty into the model while provably preserving the determinism of the underlying logical structure: under certain observations, the weighted transition matrices collapse exactly to the deterministic transition matrices of the DFA, guaranteeing the complete absence of dynamic uncertainty. The logic component is indeed composed by a WFA which by grounding the architecture in the mathematical structure of semirings, enables DWFA to unify qualitatively different inference procedures within a single architectural paradigm without requiring any structural modification between tasks.

Compared to previous neurosymbolic approaches to temporally related sequences of data, DWFA occupies a distinctive position. Unlike fuzzy NeSy architectures [3], it provides a formally grounded probabilistic interpretation of perceptual uncertainty, which, as shown, enabled efficient use of previously known results, such as likelihood-based reasoning, and provided interpretability to the model. Different from the approach proposed in [4], instead, DWFA directly encodes LTL_f temporal specifications and does not introduce any intrinsic dynamic uncertainty.

Several directions remain open for future work. The current online monitoring procedure computes compliance probabilities only up to the current time step, without providing predictive information about possible future continuations of the trace. Extending the framework to support colored automata four-valued semantics, distinguishing permanent satisfaction, permanent violation, temporary satisfaction, and temporary violation, would considerably enrich its monitoring capabilities and bring it in line with established approaches in business process verification [23]. A second natural extension concerns bidirectional inference: the procedures developed here operate strictly forward in time, and incorporating backward and bidirectional inference could provide more robust reasoning and could enable reasoning over partial or noisy observations. Finally, while this work assumes that the LTL_f specification governing the system is known a priori, relaxing this assumption by exploring settings in which the automaton structure is learned from data, along with the lines of [5].

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] A. Sheth, K. Roy, M. Gaur, Neurosymbolic artificial intelligence (why, what, and how), IEEE Intelligent Systems 38 (2023) 56–62. doi:10.1109/mis.2023.3268724.

[2] G. De Giacomo, M. Vardi, Linear temporal logic and linear dynamic logic on finite traces, IJCAI International Joint Conference on Artificial Intelligence (2013) 854–860.

[3] E. Umili, R. Capobianco, G. De Giacomo, Grounding LTLf specifications in image sequences, in: Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, 2023. doi:10.24963/kr.2023/65.

[4] N. Manginas, G. Paliouras, L. De Raedt, Nesya: Neurosymbolic automata, in: J. Kwok (Ed.), Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25, International Joint Conferences on Artificial Intelligence Organization, 2025, pp. 5950–5958. URL: https://doi.org/10.24963/ijcai.2025/662. doi:10.24963/ijcai.2025/662, main Track.

[5] E. Umili, R. Capobianco, Deepdfa: Automata learning through neural probabilistic relaxations, in: U. Endriss, F. S. Melo, K. Bach, A. J. B. Diz, J. M. Alonso-Moral, S. Barro, F. Heintz (Eds.), ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024), volume 392 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2024, pp. 1051–1058. URL: https://doi.org/10.3233/FAIA240596. doi:10.3233/FAIA240596.

[6] E. Umili, F. Argenziano, A. Barbin, R. Capobianco, Visual reward machines, in: Proceedings of the 17th International Workshop on Neural-Symbolic Learning and Reasoning, La Certosa di Pontignano, Siena, Italy, July 3-5, 2023, 2023, pp. 255–267. URL: https://ceur-ws.org/Vol-3432/paper23.pdf.

[7] E. Umili, F. Argenziano, R. Capobianco, Neural reward machines, in: U. Endriss, F. S. Melo, K. Bach, A. J. B. Diz, J. M. Alonso-Moral, S. Barro, F. Heintz (Eds.), ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024), volume 392 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2024, pp. 3055–3062. URL: https://doi.org/10.3233/FAIA240847. doi:10.3233/FAIA240847.

[8] J. E. Hopcroft, R. Motwani, J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, 2nd ed., Addison-Wesley, Reading, Massachusetts, 2001.

[9] M. Droste, W. Kuich, H. Vogler, Handbook of Weighted Automata, Springer, 2009. doi:10.1007/978-3-642-01492-5.

[10] A. Pnueli, The temporal logic of programs, 18th Annual Symposium on Foundations of Computer Science (sfcs 1977) (1977). doi:10.1109/sfcs.1977.32.

[11] M. Vardi, An automata-theoretic approach to linear temporal logic, Lecture Notes in Computer Science (1996). doi:10.1007/3-540-60915-6_6.

[12] G. De Giacomo, M. Favorito, Compositional approach to translate LTLf/LDLf into deterministic finite automata, Proceedings of the International Conference on Automated Planning and Scheduling 31 (2021) 122–130. URL: https://ojs.aaai.org/index.php/ICAPS/article/view/15954. doi:10.1609/icaps.v31i1.15954.

[13] S. Bansal, Y. Li, L. M. Tabajara, M. Y. Vardi, Hybrid compositional reasoning for reactive synthesis from finite-horizon specifications, 2020. URL: https://arxiv.org/abs/1911.08145. arXiv:1911.08145.

[14] S. Zhu, L. M. Tabajara, J. Li, G. Pu, M. Y. Vardi, Symbolic LTLf synthesis, 2017. URL: https://arxiv.org/abs/1705.08426. arXiv:1705.08426.

[15] D. C. Lay, S. R. Lay, J. J. McDonald, Linear Algebra and its Applications, fifth ed., Pearson, Boston, MA, 2016.

[16] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016. http://www.deeplearningbook.org.

[17] C. Guo, G. Pleiss, Y. Sun, K. Q. Weinberger, On calibration of modern neural networks, 2017. URL: https://arxiv.org/abs/1706.04599. arXiv:1706.04599.

[18] V. Derkinderen, R. Manhaeve, P. Zuidberg Dos Martires, L. De Raedt, Semirings for probabilistic and neuro-symbolic logic programming, International Journal of Approximate Reasoning 171 (2024) 109130. URL: http://dx.doi.org/10.1016/j.ijar.2024.109130. doi:10.1016/j.ijar.2024.109130.

[19] G. De Giacomo, R. Masellis, M. Grasso, F. Maggi, M. Montali, Monitoring business metaconstraints based on ltl and ldl for finite traces, 2014. doi:10.1007/978-3-319-10172-9_1.

[20] F. M. Maggi, M. Montali, R. Peñaloza, A. Alman, Extending temporal business constraints with uncertainty, in: D. Fahland, C. Ghidini, J. Becker, M. Dumas (Eds.), Business Process Management, Springer International Publishing, Cham, 2020, pp. 35–54.

[21] F. M. Maggi, M. Westergaard, M. Montali, W. M. P. van der Aalst, Runtime verification of ltl-based declarative process models, in: S. Khurshid, K. Sen (Eds.), Runtime Verification, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 131–146.

[22] G. Forney, The viterbi algorithm, Proceedings of the IEEE 61 (1973) 268–278. doi:10.1109/PROC.1973.9030.

[23] F. M. Maggi, M. Montali, M. Westergaard, W. M. P. van der Aalst, Monitoring business constraints with linear temporal logic: An approach based on colored automata, in: S. Rinderle-Ma, F. Toumani, K. Wolf (Eds.), Business Process Management, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 132–147.