

# CompAct Nets - A unifying tensor network architecture for probabilistic and logical reasoning

## Author One

Department of Statistics  
University of Washington  
Seattle, WA 98195-4322, USA

ONE@STAT.WASHINGTON.EDU

## Author Two

Division of Computer Science  
University of California  
Berkeley, CA 94720-1776, USA

TWO@CS.BERKELEY.EDU

**Editor:** My editor

## Abstract

We introduce Computation-Activation Networks (CompAct Nets), a novel architecture for tensor networks, which is adapted to represent propositional formulas and exponential distributions.

**Keywords:** Tensor Networks, Neuro-Symbolic AI

## 1 Introduction

Modern artificial intelligence is dominated by large-scale neural models that excel at pattern recognition but mostly remain black-box-solvers. Therefore, reliability and explainability are two main concerns when integrating these architecture into safety-critical processes. In contrast, classical symbolic approaches offer explicit logical structures and human-readable inference but can not handle uncertainty or scale to complex real-world data. Probabilistic models improved uncertainty handling but at the cost of explainability. Bridging these paradigms achieving both expressive architectures and transparent reasoning defines the central goal of *Neuro-Symbolic AI*, which seeks methods that combine the structural clarity of logic with the adaptability of neural computation within a single, mathematically coherent framework.

A central goal is to achieve *intrinsic explainability* rather than post-hoc interpretation. In conventional neural models, there are various ways to interpret a model after it has been trained, e.g. based on analyzing how changing input features influence the models prediction or based on fitting simpler surrogate models Barredo Arrieta et al. (2020); Lipton (2017). The proposed framework encodes symbolic relations that remain directly readable. Explainability is thus not added after training but built into the architecture itself.

The *tnreason framework* proposes tensor networks as a unifying mathematical foundation for reasoning and probabilistic inference. Tensor networks factor complex systems into interconnected logical and probabilistic components.

In this unifying mathematical framework, logical formulas corresponding to boolean tensors and probabilistic distributions corresponding to non-negative real tensors are combined.

Both can be manipulated through the same algebra of tensor contraction. This abstraction eliminates the traditional divide between symbolic and numerical representations: logical inference and probabilistic computations become different instances of the same underlying operation on structured tensors.

The *computation-activation architecture* organizes reasoning into two complementary tensor substructures. The computation network encodes the structural relations of a problem, such as logical dependencies or sufficient statistics, while the activation network assigns semantic or numerical values to these structures, representing truth assignments or probabilities. Their interaction defines a reasoning process as a tensor contraction between structure and activation. Logical inference emerges when activations are boolean, probabilistic inference when they are real-valued, and hybrid reasoning when both coexist within a shared tensor representation.

The logical tradition of artificial intelligence is motivated by the resemblance of human thought in logics McCarthy. Historic approaches to artificial intelligence have focused on models by vast knowledge bases and inference by logical reasoning. The main problem hindering the success of this approach is the inability of classical first-order logic to handle uncertainty of information, as present in realistic scenarios.

Towards extending the practical usage of logics, the field of Statistical Relational AI Nickel et al.; Getoor and Taskar studies statistical models of logical relations. This directly treats uncertainty and therefore unifies logics with statistical approaches. These aims have more recently reframed as neuro-symbolic AI Hochreiter; Sarker et al.; Colelough and Regli (2025), with close relations to statistical relational AI Marra et al.. Neuro-symbolic AI focuses on the unification of the neural and the symbolic paradigm Garcez et al., where early approaches are Towell and Shavlik; Avila Garcez and Zaverucha. While the symbolic paradigm is roughly understood as human understandable reasoning in formal logics, the neural paradigm is the computational benefit of decomposing a model into layers. These decompositions provide both expressive and efficiently inferable model architectures. While modern black-box AI focuses on large neural networks, whose size prevents human understanding of the inference process, neuro-symbolic AI aims at a re-implementation of the symbolic paradigm into such architectures.

Tensor networks have emerged as a highly efficient mathematical framework for handling data in high-dimensional spaces, effectively circumventing the "curse of dimensionality" that typically plagues grid-based methods Hackbusch. By decomposing high-order tensors into networks of low-rank components, these structures reduce the storage and computational complexity from exponential to polynomial with respect to the dimension Oseledets (2011); Hackbusch and Kühn (2009); Hitchcock (1927).

Historically rooted in quantum many-body physics White (1992), this framework found its first major success with Matrix Product States (MPS), originally developed to efficiently capture the quantum dynamics and ground states of one-dimensional spin chains Affleck et al. (1987). This format remains a standard tool in the field, with recent contributions refining it for tasks such as large-scale stochastic simulations and variational circuit operations Sander et al., 2025). To address the topological constraints of MPS, the landscape

of architectures was subsequently expanded to include Projected Entangled Pair States (PEPS) for two-dimensional lattices and the Multi-scale Entanglement Renormalization Ansatz (MERA), which utilizes a hierarchical geometry to represent scale-invariant critical systems and has recently been adapted for simulating quantum systems Berezutskii et al. (2025); Orús (2019).

Beyond the quantum realm, these formats have been successfully adapted to applied mathematics, particularly for solving high-dimensional parametric PDEs, sampling problems, modeling complex continuous fields and learning dynamical laws Hagemann et al. (2025); Eigel et al. (2017); Goeßmann et al.. Furthermore, they exhibit properties helpful for handling these high-dimensional spaces, such as restricted isometry properties Goeßmann. Recent advancements have demonstrated the efficacy of these methods in capturing multiscale phenomena in fluid dynamics and turbulence, proving that the tensor network formalism offers a robust alternative to classical numerical schemes Gourianov et al. (2025).

Most significantly for the present work, we exploit the algebraic flexibility of tensor networks to bridge the gap between continuous numerical representation and discrete symbolic reasoning. By interpreting tensor contractions as logical operations within a basis calculus we can rigorously map propositional logic onto the linear-algebraic substrate of tensor networks. As shown in Goessmann (2025), this very versatile and flexible framework can be applied to unify logical and probabilistic modeling, enabling a single architecture to perform exact symbolic inference while retaining the efficient learnability of high-dimensional neural representations.

## 1.1 Related Works

The unification of symbolic and probabilistic approaches to interpretable model architectures has been a long-standing aim. Probabilistic graphical models Pearl; Koller and Friedman are means to encode variable independences in graphs and are specific instances of exponential families Wainwright and Jordan. Markov Logic Networks Richardson and Domingos are specific instances of exponential families where also the dependencies are explicitly encoded in logical formulas. Further approaches treat uncertainties as generalized truth values ?.

Tensor Networks have recently gained interest as a unifying language for AI, framed by Logical Tensor Networks Badreddine et al. and Tensor Logic Domingos. Different to the approaches therein we do not require non-linear transforms of tensors. Further, the MeLoCoToN approach Ali applies tensor network architectures similar to Computation-Activation Networks in combinatorial optimization problems.

## 1.2 Structure of the paper

The paper is organized as follows. Section 2 introduces the basic notation for categorical variables, tensors, and tensor networks, establishing the formal framework on which all subsequent reasoning structures are defined. Section 3 develops the probabilistic representation of reasoning through soft activation, showing how exponential-family distributions can be expressed as tensor networks based on independence assumptions and sufficient statistics. Section 4 turns to hard activation, formulating propositional logic within the same tensor framework and demonstrating how logical inference, entailment, and knowledge bases can

be represented by boolean tensors and contractions. Section 5 unifies these two perspectives in the concept of Hybrid Logic Networks, which integrate hard logical constraints with soft probabilistic activations, thereby forming the core of the computation–activation architecture. The paper concludes with algorithmic considerations in section ?? and examples in section 6 that illustrate the expressive power and interpretability of this unified tensor-based reasoning approach.

## 2 Notation and Basic Concepts

We first, introduce the basic architecture and later show how the probabilistic and logical framework are covered. This is done in multiple steps. First, the considered variables and tensors are explained. Followed by the explanation of tensor calculations, mainly tensor products and contractions. Finally, the main architecture, the *Computation-Activation Network*, is defined.

### 2.1 Tensors

Tensors are multiway arrays and a generalization of vectors and matrices to higher orders. We will first provide a formal definition as real maps from index sets enumerating the coordinates of vectors, matrices and larger order tensors.

**Definition 1 (Tensor)** For  $k \in [d]$ , let  $m_k \in \mathbb{N}$  and let  $X_k$  be categorical variables with values in  $[m_k]$ . A tensor  $\tau[X_0, \dots, X_{d-1}]$  of order  $d$  and with leg dimensions  $m_0, \dots, m_{d-1}$  is defined through its coordinates

$$\tau[X_0 = x_0, \dots, X_{d-1} = x_{d-1}] \in \mathbb{R}$$

for index tuples

$$x_0, \dots, x_{d-1} \in \bigtimes_{k \in [d]} [m_k].$$

Tensors  $\tau[X_0, \dots, X_{d-1}]$ , also denoted by  $\tau[X_{[d]}]$ , are elements of the tensor space

$$\bigotimes_{k \in [d]} \mathbb{R}^{m_k},$$

which is a linear space, enriched with the operations of coordinate wise summation and scalar multiplication. We call a tensor  $\tau[X_{[d]}]$  boolean, when  $\text{im}(\tau) \subset [2]$ , i.e. all coordinates are either 0 or 1.

This notation of tensors opposed to its notation through ordered indices as common in tensor calculus, facilitates writing down contractions along individual legs and other operations. Occasionally, when the categorical variables of a tensor are clear from the context, we will omit the notation of the variables.

**Example 1 (Trivial Tensor)** *The trivial tensor*

$$\mathbb{I}[X_{[d]}] \in \bigotimes_{k \in [d]} \mathbb{R}^{m_k}$$

is defined by all coordinates being 1, that is for all  $x_0, \dots, x_{d-1} \in \times_{k \in [d]} [m_k]$

$$\mathbb{I}[X_{[d]} = x_{[d]}] = 1.$$

We are now ready to provide the link between tensors and states of systems with factored representations. To this end, we define the one-hot encoding of a state, which is a bijection between the states and the basis elements of a tensor space.

**Definition 2 (One-hot encodings to Atomic Representations)** *Given an atomic system described by the categorical variable  $X$ , we define for each  $x \in [m]$  the basis vector  $\epsilon_x[X]$  by the coordinates*

$$\epsilon_x[X = \tilde{x}] = \begin{cases} 1 & \text{if } x = \tilde{x} \\ 0 & \text{else.} \end{cases} \quad (1)$$

The one-hot encoding of states  $x \in [m]$  of the atomic system described by the categorical variable  $X$  is the map  $\epsilon : [m] \rightarrow \mathbb{R}^m$  which maps  $x \in [m]$  to the basis vectors  $\epsilon_x[X]$ .

The basis vectors  $\epsilon_x[X]$  are tensors of order 1 and leg dimension  $m$  of the structure

$$\epsilon_x[X] = [0 \quad \dots \quad 0 \quad 1 \quad 0 \quad \dots \quad 0]^T, \quad (2)$$

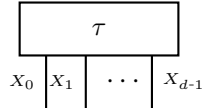
where the 1 is at the  $x$ th coordinate of the vector.

## 2.2 Contractions and Tensor Networks

Contractions are the central manipulation operation on sets of tensors. To introduce them, a graphical illustration of sets of tensors is explained, which we also call tensor networks.

### 2.2.1 GRAPHICAL ILLUSTRATION

We will use the standard visualization of tensors, where they are represented by blocks with lines depicting the axes of the tensor blocks and each axis is assigned with a categorical variable  $X_k$ , or sometimes their index or dimension.



This depiction scheme has been established in the literature as wiring diagrams (see Landsberg and dates back at least to the work Penrose. Along this line, we represent vectors by blocks with one leg. The basis vectors being one-hot encodings of states are in this scheme represented by

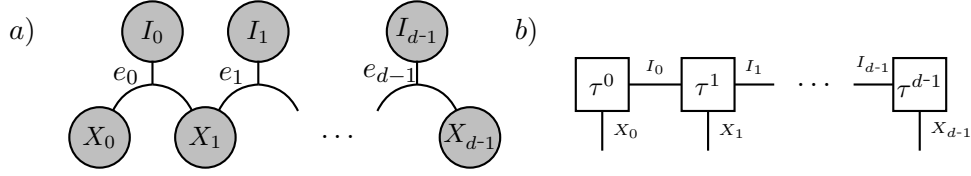


Figure 1: Hypergraph to a TT format. a) Node-centric design. b) Corresponding tensor-network on the edges of the hypergraph.



Assigning  $x$  to the categorical variable  $X$  will retrieve the  $x$ th coordinate (with value 1), whereas all other assignments will retrieve the coordinate values 0. Drawing on the interpretation of tensors by hyperedges we can continue with the definition of tensor networks.

**Definition 3 (Tensor Network)** Let  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  be a hypergraph with nodes decorated by categorical variables  $X_v$  with dimensions  $m_v \in \mathbb{N}$  and hyperedges  $e \in \mathcal{E}$  decorated by core tensors

$$\tau^e[X_e] \in \bigotimes_{v \in e} \mathbb{R}^{m_v},$$

where we denote by  $X_e$  the set of categorical variables  $X_v$  with  $v \in e$ . Then we call the set

$$\tau^{\mathcal{G}}[X_{\mathcal{V}}] = \{\tau^e[X_e] : e \in \mathcal{E}\}$$

the Tensor Network of the decorated hypergraph  $\mathcal{G}$ . The set of tensor networks on  $\mathcal{G}$ , such that all tensors have non-negative coordinates, is denoted by  $\mathcal{T}^{\mathcal{G}}$ .

**Example 2 (The TT format)** The Tensor-Train (TT) (see Oseledets (2011)) format corresponds in our notation with a hypergraph (see Figure 2)

- Nodes by  $X_{[d]}$  and hidden variables  $I_{[d-1]}$ , each decorated by a dimension  $m_{[d]}$  and  $n_{[d-1]}$
- Edges by  $\{e_0 = (X_0, I_0)\} \cup \{e_k = (I_{k-1}, X_k, I_{k+1}) : k \in \{1, \dots, d-2\}\} \cup \{e_{d-1} = (I_{d-2}, X_{d-1})\}$  each decorated by a tensor of order 3 (respectively 2 for  $k \in \{0, d-1\}$ ).

**Example 3 (The CP format)** The Candecomp-Parafac (CP) (?) tensor format corresponds in our notation with a hypergraph (see Figure 3)

- Nodes by  $X_{[d]}$  and a single hidden variable  $I$ , decorated by dimensions  $m_{[d]}$  and  $n$ .
- Edges by

$$\{e_k = (X_k, I) : k \in [d]\}$$

each decorated by a matrix  $\tau^{e_k}[X_k, I]$ .

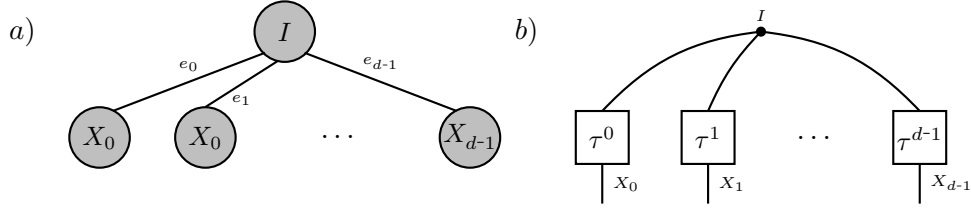


Figure 2: Hypergraph to a CP format. a) Node-centric design. b) Corresponding tensor-network on the edges of the hypergraph.

### 2.2.2 TENSOR PRODUCT

Let us now exploit the developed graphical representations to define contractions of tensor networks. The simplest contraction is the tensor product, which maps a pair of two tensors with distinct variables onto a third tensor and has an interpretation by coordinate wise products. Such a contraction corresponds with a tensor network of two tensors with disjoint variables.

**Definition 4 (Tensor Product)** *Let there be two tensors*

$$\tau [X_{[d]}] \in \bigotimes_{k \in [d]} \mathbb{R}^{m_k} \quad \text{and} \quad \tilde{\tau} [Y_{[p]}] \in \bigotimes_{l \in [p]} \mathbb{R}^{m_l}$$

*with different categorical variables assigned to its axes. Then their tensor product is the map*

$$\langle \tau [X_{[d]}], \tilde{\tau} [Y_{[p]}] \rangle_{[X_{[d]}, Y_{[p]}]} \in \left( \bigotimes_{k \in [d]} \mathbb{R}^{m_k} \right) \otimes \left( \bigotimes_{l \in [p]} \mathbb{R}^{m_l} \right)$$

*defined coordinatewise for tuples of  $x_0, \dots, x_{d-1} \in \times_{k \in [d]} [m_k]$  and  $y_0, \dots, y_{p-1} \in \times_{l \in [p]} [m_l]$  as*

$$\begin{aligned} & \langle \tau [X_{[d]}], \tilde{\tau} [Y_{[p]}] \rangle_{[X_0=x_0, \dots, X_{d-1}=x_{d-1}, Y_0=y_0, \dots, Y_{p-1}=y_{p-1}]} \\ & := \tau [X_0 = x_0, \dots, X_{d-1} = x_{d-1}] \cdot \tilde{\tau} [Y_0 = y_0, \dots, Y_{p-1} = y_{p-1}]. \end{aligned}$$

### 2.3 Generic Contractions

Contractions of Tensor Networks  $\tau^{\mathcal{G}}$  are operations to retrieve single tensors by summing products of tensors in a network over common indices. We will define contractions formally by specifying just the indices not to be summed over.

When some of the variables are not appearing as leg variables, we define the contraction as being a tensor product with the trivial tensor  $\mathbb{I}$  carrying the legs of the missing variables.

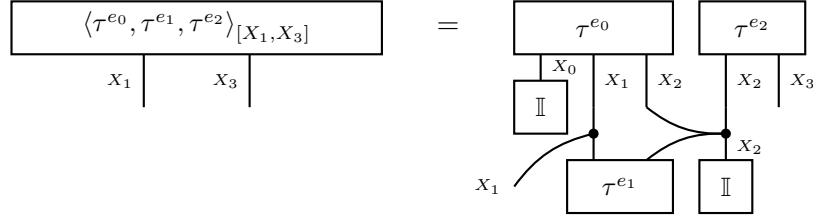


Figure 3: Example of a tensor network contraction of all but the variables  $X_1, X_3$ . Contraction of variables can always be depicted by closing the open legs with trivial tensors  $\mathbb{I}$  performing index sums.

**Definition 5** Let  $\tau^{\mathcal{G}}$  be a tensor network on a decorated hypergraph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ . For any subset  $\tilde{\mathcal{V}} \subset \mathcal{V}$  we define the contraction to be the tensor (for an example see Figure 3)

$$\langle \tau^{\mathcal{G}} \rangle_{[X_{\tilde{\mathcal{V}}}] \in \bigotimes_{v \in \tilde{\mathcal{V}}} \mathbb{R}^{m_v}} \quad (3)$$

defined coordinatewise by the sum

$$\langle \tau^{\mathcal{G}} \rangle_{[X_{\tilde{\mathcal{V}}}=x_{\tilde{\mathcal{V}}}]} = \sum_{x_{\mathcal{V}/\tilde{\mathcal{V}}} \in \times_{v \in \mathcal{V}/\tilde{\mathcal{V}}} [m_v]} \left( \prod_{e \in \mathcal{E}} \tau^e [X_e = x_e] \right). \quad (4)$$

We call  $X_{\tilde{\mathcal{V}}}$  the open variables of the contraction.

To ease notation, we will often omit the set notation by brackets  $\{\cdot\}$  and specify the tensors to be contracted with the delimiter “,” (see e.g. Example ??).

## 2.4 Directed Tensors and Normalizations

Directionality represents constraints on the structure of tensors, namely that the sum over outgoing trivializes the tensor.

**Definition 6** A Tensor

$$\tau [X_{\mathcal{V}}] \in \bigotimes_{v \in \mathcal{V}} \mathbb{R}^{m_v}$$

is said to be directed with incoming variables  $\mathcal{V}^{\text{in}}$  and outgoing variables  $\mathcal{V}^{\text{out}}$ , where  $\mathcal{V} = \mathcal{V}^{\text{in}} \dot{\cup} \mathcal{V}^{\text{out}}$ , when

$$\langle \tau \rangle_{[X_{\mathcal{V}^{\text{in}}}] = \mathbb{I} [X_{\mathcal{V}^{\text{in}}}]$$

where  $\mathbb{I} [X_{\mathcal{V}^{\text{in}}}]$  denoted the trivial tensor in  $\bigotimes_{v \in \mathcal{V}^{\text{in}}} \mathbb{R}^{m_v}$  which coordinates are all 1.

By the normalization operation, tensors are turned into directed tensors.

**Definition 7** A tensor  $\tau[X_{\mathcal{V}}]$  is said to be normalizable on  $\mathcal{V}^{\text{in}} \subset \mathcal{V}$ , if for any  $x_{\mathcal{V}^{\text{in}}} \in \times_{v \in \mathcal{V}^{\text{in}}} [m_v]$  we have

$$\left\langle \tau[X_{\mathcal{V}}], \epsilon_{x_{\mathcal{V}^{\text{in}}}}[X_{\mathcal{V}^{\text{in}}}] \right\rangle_{[\emptyset]} > 0.$$

The normalization of an on  $\mathcal{V}^{\text{in}} \subset \mathcal{V}$  normalizable tensor is the tensor

$$\langle \tau[X_{\mathcal{V}}] \rangle_{[X_{\mathcal{V}^{\text{out}}}|X_{\mathcal{V}^{\text{in}}}] = \sum_{x_{\mathcal{V}^{\text{in}}} \in \times_{v \in \mathcal{V}^{\text{in}}} [m_v]} \epsilon_{x_{\mathcal{V}^{\text{in}}}}[X_{\mathcal{V}^{\text{in}}}] \otimes \frac{\left\langle \tau[X_{\mathcal{V}}], \epsilon_{x_{\mathcal{V}^{\text{in}}}}[X_{\mathcal{V}^{\text{in}}}] \right\rangle_{[X_{\mathcal{V}^{\text{out}}}]}}{\left\langle \tau[X_{\mathcal{V}}], \epsilon_{x_{\mathcal{V}^{\text{in}}}}[X_{\mathcal{V}^{\text{in}}}] \right\rangle_{[\emptyset]}}$$

where  $\mathcal{V}^{\text{out}} = \mathcal{V}/\mathcal{V}^{\text{in}}$ .

In our graphical tensor notation, we depict directed tensors by directed hyperedges (a), which are decorated by directed tensors (b), for example:



## 2.5 Function encoding and Computation-Activation Networks

Let us now encode functions between the state sets of systems in factored representation.

**Definition 8 (Basis encoding of maps between state sets)** Let there be two systems with factored representations by variables  $X_{[d]}$  and  $Y_{[p]}$ , and a map

$$q : \times_{k \in [d]} [m_k] \rightarrow \times_{l \in [p]} [m_l]$$

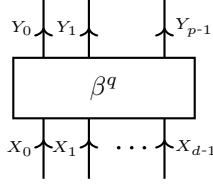
between these state sets. Then the basis encoding of  $q$  is a tensor

$$\beta^q[Y_{[p]}, X_{[d]}] \in \left( \bigotimes_{l \in [p]} \mathbb{R}^{m_l} \right) \otimes \left( \bigotimes_{k \in [d]} \mathbb{R}^{m_k} \right)$$

defined by

$$\beta^q[Y_{[p]}, X_{[d]}] = \sum_{x_0, \dots, x_{d-1} \in \times_{k \in [d]} [m_k]} \epsilon_{q(x_{[d]})}[Y_{[p]}] \otimes \epsilon_{x_{[d]}}[X_{[d]}].$$

Basis encodings are directed tensors (see (Goessmann, 2025, Theorem 14.10)) and are thus depicted as decorations of directed edges in hypergraphs: [Explain the arrows.](#) [Haven't introduced directed graphs yet.](#)



The entries of the basis encoding are defined by

$$\beta^q [Y_{[p]} = y_{[p]}, X_{[d]} = x_{[d]}] = \begin{cases} 1 & \text{if } q(x_{[d]}) = y_{[p]} \\ 0 & \text{else} \end{cases}$$

Based on these concepts the main architecture can be defined.

**Definition 9 (Computation-Activation Network (CompAct Nets))** *Given a statistic  $\mathcal{S} : \times_{k \in [d]} [m_k] \rightarrow \mathbb{R}^p$ , and a hypergraph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with nodes  $[p] \subset \mathcal{V}$  containing the image coordinates of  $\mathcal{S}$ , we define the by  $\mathcal{S}$  computable and by  $\mathcal{G}$  activated family of distributions by*

$$\Lambda^{\mathcal{S}, \mathcal{G}} = \left\{ \left\langle \beta^{\mathcal{S}} [Y_{[p]}, X_{[d]}], \langle \xi \rangle_{[Y_{[p]}]} \right\rangle_{[X_{[d]} | \emptyset]} : \xi [Y_{\mathcal{V}}] \in \mathcal{T}^{\mathcal{G}} \right\}.$$

We refer to any member  $\mathbb{P} [X_{[d]}] \in \Lambda^{\mathcal{S}, \mathcal{G}}$  as a *Computation-Activation Network*.

To represent a Computation-Activation Network two tensor networks are needed:  $\beta^{\mathcal{S}}$  to represent the basis encoding of the sufficient statistic (also called *computation network*) and  $\xi$  to represent the tensor to contract with (also called *activation network*).

### 3 Probabilistic representation: Exponential Families through soft activation

We begin with defining tensors representing the probabilistic side of the desired structure unifying probabilistic and logical reasoning.

After introducing probability distributions as tensors we derive tensor network decompositions based on conditional independencies (applying the Hammersley-Clifford theorem Clifford and Hammersley) to motivate graphical models.

It is known that probabilistic graphical models are dual to tensor networks Robeva and Seigal; Glasser et al.. By our hypergraph based definition of tensor networks, markov networks are equivalent to tensor networks of positive tensors.

#### 3.1 Basic concepts

Distributions  $\mathbb{P}$  over a discrete state space can be represented by tensors, where each entry corresponds to the probability of a corresponding state. The joint probability distribution for a set of categorical variables as in definition ?? is defined here.

**Definition 10 (Joint Probability Distribution)** *Let there be for each  $k \in [d]$  a categorical variable  $X_k$  taking values in  $[m_k]$ . A joint probability distribution of these categorical variables is a function*

$$\mathbb{P}[X_{[d]}] : \prod_{k \in [d]} [m_k] \rightarrow \mathbb{R}$$

*which is non-negative, that is for any  $x_{[d]} \in \prod_{k \in [d]} [m_k]$  it holds  $\mathbb{P}[X_{[d]} = x_{[d]}] \geq 0$ , and which is normalized, that is  $\langle \mathbb{P}[X_{[d]}] \rangle_{[\emptyset]} = 1$ .*

**Example 4 Coin toss** *Consider two coin tosses  $X_0, X_1 \in \{0, 1\}$  ( $1=\text{heads}$ ). With  $p \in [0, 1]$  being the probability of heads. Then the probability for each toss and the joint probability have the form*

$$\mathbb{P}[X_0] = \begin{bmatrix} \mathbb{P}[X_0 = 0] \\ \mathbb{P}[X_0 = 1] \end{bmatrix} = \begin{bmatrix} 1-p \\ p \end{bmatrix}, \quad \mathbb{P}[X_{[2]}] = \begin{bmatrix} (1-p)^2 & (1-p)p \\ p(1-p) & p^2 \end{bmatrix}$$

In most of the analysis, we assume that every state in the space is assigned a positive probability. We say, a distribution  $\mathbb{P}[X_{[d]}]$  is *positive* if  $\mathbb{P}[X_{[d]} = x_{[d]}] > 0$  for every configuration  $x_{[d]}$ . Note that in this tensor centered notation the calculation of marginal distributions reduces to contracting the open legs, not considered in the marginal with tensors  $\mathbb{I}$  only containing ones, see (Goessmann, 2025, Section 2.1.3).

$$\begin{array}{c} \boxed{\mathbb{P}[X_0]} \\ \downarrow x_0 \end{array} = \begin{array}{c} \boxed{\mathbb{P}[X_0, X_1]} \\ \downarrow x_0 \quad \downarrow x_1 \\ \boxed{\mathbb{I}} \end{array}$$

The number of coordinates in a tensor representation of probability distributions is the product

$$\prod_{k \in [d]} m_k,$$

and therefore scales exponentially in the number of coordinates. To find efficient representation schemes of probability distributions by tensor networks, we need to exploit additional properties of the distribution, such as Markov chain properties or independence. We further explore one property based on sufficient statistics.

### 3.2 The Independence mechanism: Graphical Model Factorization

Recall Def. 9. Given a statistic  $S : \prod_{k \in [d]} [m_k] \rightarrow \mathbb{R}^p$  and a hypergraph  $G = (V, E)$  on the image coordinates  $Y[p]$ , the by  $S$  computable and by  $G$  activated family of distributions has the form

$$P[X[d]] \in \Lambda_{S,G} = \langle \beta_S[Y[p], X[d]], \xi[Y[p]] \rangle_{[X[d]|\emptyset]}.$$

For graphical models we take the *identity statistic*

$$S_{\text{id}}(x[d]) = x[d],$$

so that the image coordinates coincide with the variables,  $Y[p] = X[d]$ , and there are no non-trivial computation cores. The associated basis encoding is just the identity tensor between  $Y[d]$  and  $X[d]$ ,

$$\beta_{S_{\text{id}}}[Y[d] = y[d], X[d] = x[d]] = \mathbf{1}_{\{y[d]=x[d]\}},$$

and therefore, for any activation tensor  $\xi[Y[d]]$  we obtain

$$\begin{aligned} P[X[d]] &= \langle \beta_{S_{\text{id}}}[Y[d], X[d]], \xi[Y[d]] \rangle_{[X[d]|\emptyset]} \\ &= \xi[X[d]]. \end{aligned}$$

In other words, in the graphical-model case the activation tensor coincides with the joint distribution tensor. In this setting, structural properties of the distribution such as (conditional) independences can be read off as algebraic factorization patterns of the activation (and hence joint) tensor.

Independence leads to severe sparsifications of conditional probabilities and is therefore the key assumption to gain sparse decompositions of probability distributions. Before showing such decomposition schemes, we first provide a coordinatewise definition of independent variables.

**Definition 11 (Independence)** *We say that  $X_0$  is independent of  $X_1$  with respect to a distribution  $\mathbb{P}[X_0, X_1]$ , if for any values  $x_0 \in [m_0]$  and  $x_1 \in [m_1]$  the distribution satisfies*

$$\mathbb{P}[X_0 = x_0, X_1 = x_1] = \mathbb{P}[X_0 = x_0] \cdot \mathbb{P}[X_1 = x_1].$$

*In this case we denote  $(X_0 \perp X_1)$ .*

This can be transferred to a contraction equation of tensor networks:

**Theorem 12 (Independence Criterion as a Contraction Equation)** *The variable  $X_0$  is independent from  $X_1$  with respect to a probability distribution  $\mathbb{P}[X_0, X_1]$ , if and only if*

$$\mathbb{P}[X_0, X_1] = \left\langle \langle \mathbb{P}[X_0, X_1] \rangle_{[X_0]}, \langle \mathbb{P}[X_0, X_1] \rangle_{[X_1]} \right\rangle_{[X_0, X_1]}.$$

The proof of this theorem can be found in (Goessmann, 2025, Theorem 2.12).

In the graphical-model CAN with identity statistic, the joint distribution  $\mathbb{P}[X_0, X_1]$  is represented by the activation tensor  $\xi[X_0, X_1]$ . By Thm. 12, independence of  $X_0$  and  $X_1$  is equivalent to the factorization

$$\mathbb{P}[X_0, X_1] = \mathbb{P}[X_0] \mathbb{P}[X_1],$$

which in activation form reads

$$\xi[X_0, X_1] = \xi[X_0] \otimes \xi[X_1].$$

Thus, independence appears directly as a tensor-product decomposition of the activation (and hence joint) tensor into two unary activation cores, already implying a substantial reduction in degrees of freedom.

Two jointly distributed variables are by 12 independent, if and only if their joint distribution  $\mathbb{P}[X_0, X_1]$  is the tensor product of marginal probabilities. Using tensor network diagrams we depict this property by

$$\begin{array}{c} \boxed{\mathbb{P}[X_0, X_1]} \\ \downarrow x_0 \quad \downarrow x_1 \end{array} = \begin{array}{c} \boxed{\mathbb{P}[X_0, X_1]} \\ \downarrow x_0 \quad \downarrow x_1 \end{array} \otimes \begin{array}{c} \boxed{\mathbb{P}[X_0, X_1]} \\ \downarrow x_0 \quad \downarrow x_1 \end{array} = \begin{array}{c} \boxed{\mathbb{P}[X_0]} \\ \downarrow x_0 \end{array} \otimes \begin{array}{c} \boxed{\mathbb{P}[X_1]} \\ \downarrow x_1 \end{array} .$$

Let us notice, that the assumption of independence reduces the degrees of freedom from  $m_0 \cdot m_1 - 1$  to  $(m_0 - 1) + (m_1 - 1)$ . The decomposition into marginal distributions furthermore exploits this reduced freedom and provides an efficient storage. Having a joint distribution of multiple variables, which disjoint subsets are independent, we can iteratively apply the decomposition scheme. As a result, we can reduce the scaling of the degrees of freedom from exponential to linear by the assumption of independence.

Independence is, as we observed, a strong assumption, which is often too restrictive. Conditional independence instead is a less demanding assumption, which still implies efficient tensor network decompositions schemes. We introduce conditional independence as independence of variables with respect to conditional distributions.

**Definition 13 (Conditional Independence)** *Given a joint distribution of variables  $X_0$ ,  $X_1$  and  $X_2$ , such that  $\mathbb{P}[X_2]$  is positive. We say that  $X_0$  is independent of  $X_1$  conditioned on  $X_2$  if for any states  $x_0 \in [m_0]$ ,  $x_1 \in [m_1]$  and  $x_2 \in [m_2]$*

$$\mathbb{P}[X_0 = x_0, X_1 = x_1 | X_2 = x_2] = \mathbb{P}[X_0 = x_0 | X_2 = x_2] \cdot \mathbb{P}[X_1 = x_1 | X_2 = x_2] .$$

In this case we denote  $(X_0 \perp X_1) | X_2$ .

There are different ways to decompose a probability distribution into hopefully smaller parts, which can be represented more efficiently.

Conditional independence stated in Def. 13 has a close connection with independence stated in Def. 11. To be more precise,  $X_0$  is independent of  $X_1$  conditioned on  $X_2$ , if and only if  $X_0$  is independent of  $X_1$  with respect to any slice  $\mathbb{P}[X_0, X_1 | X_2 = x_2]$  of the conditional distribution  $\mathbb{P}[X_0, X_1 | X_2]$ . We further find a decomposition criterion for conditional independence. Since conditional independence can be regarded as a property of conditional probabilities, this decomposition criterion also involves conditional probabilities.

**Theorem 14 (Conditional Independence as a Contraction Equation)** *Given a distribution  $\mathbb{P}$  of variables  $X_0$ ,  $X_1$  and  $X_2$ , the variable  $X_0$  is independent of  $X_1$  conditioned on  $X_2$ , if and only if the equation*

$$\mathbb{P}[X_0, X_1 | X_2] = \langle \mathbb{P}[X_0 | X_2], \mathbb{P}[X_1 | X_2] \rangle_{[X_0, X_1, X_2]}$$

holds.

We can further exploit conditional independence to find tensor network decompositions of probabilities, as we show as the next corollary.

**Corollary 15** *If and only if  $X_0$  is independent of  $X_1$  conditioned on  $X_2$  the probability distribution  $\mathbb{P}$  satisfies (see Figure 4)*

$$\mathbb{P}[X_0, X_1, X_2] = \langle \mathbb{P}[X_0 | X_2], \mathbb{P}[X_1 | X_2], \mathbb{P}[X_2] \rangle_{[X_0, X_1, X_2]} .$$

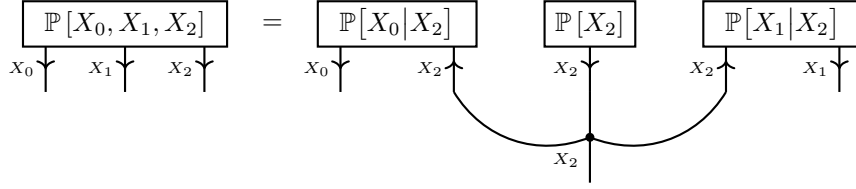


Figure 4: Diagrammatic visualization of the contraction equation in Cor. 15. Conditional independence of  $X_0$  and  $X_1$  given  $X_2$  holds if the contraction on the right side is equal to the probability tensor on the left side.

By Thm. 14 and Cor. 15, the conditional independence  $X_0 \perp X_1 \mid X_2$  is equivalent to the factorization

$$\mathbb{P}[X_0, X_1, X_2] = \langle \mathbb{P}[X_0 \mid X_2], \mathbb{P}[X_1 \mid X_2], \mathbb{P}[X_2] \rangle [X_0, X_1, X_2],$$

which is precisely the three-node graphical model with edges  $(X_0, X_2)$  and  $(X_1, X_2)$  depicted in Figure 4. In the graphical-model CAN with identity statistic, the joint activation tensor  $\xi[X_0, X_1, X_2]$  thus decomposes into three smaller cores: a unary core on  $X_2$  and two binary cores on  $(X_0, X_2)$  and  $(X_1, X_2)$ . This conditional-independence pattern is the basic local building block that is generalized in Markov networks, which we define in the following.

We define them based on hypergraphs, to establish a direct connection with tensor network decorating the hypergraph. In a more canonical way, Markov Networks are instead defined by graphs, where instead of the edges the cliques are decorated by factor tensors (see for example Koller and Friedman).

**Definition 16 (Markov Network)** Let  $\tau^{\mathcal{G}}$  be a tensor network of non-negative tensors decorating a hypergraph  $\mathcal{G}$ . Then the Markov Network  $\mathbb{P}^{\mathcal{G}}$  to  $\tau^{\mathcal{G}}$  is the probability distribution of  $X_{\mathcal{V}}$  defined by the tensor

$$\mathbb{P}^{\mathcal{G}}[X_{\mathcal{V}}] = \frac{\langle \{\tau^e : e \in \mathcal{E}\} \rangle_{[X_{\mathcal{V}}]}}{\langle \{\tau^e : e \in \mathcal{E}\} \rangle_{[\emptyset]}} = \langle \tau^{\mathcal{G}} \rangle_{[X_{\mathcal{V}}|\emptyset]}.$$

We call the denominator

$$\mathcal{Z}(\tau^{\mathcal{G}}) = \langle \{\tau^e : e \in \mathcal{E}\} \rangle_{[\emptyset]}$$

the partition function of the tensor network  $\tau^{\mathcal{G}}$ .

We can interpret the factors  $\tau_e[X_e]$  as activation cores placed on the cliques  $e$  of the graph. The global activation tensor (and hence the joint distribution) is obtained by contracting this activation network and normalizing by its partition function.

We call  $\mathbb{P}_G[\cdot]$  *positive* if  $\mathbb{P}_G[x_{\mathcal{V}}] > 0$  for all configurations  $x_{\mathcal{V}}$ . The marginalization of a Markov Network to  $\tau^{\mathcal{G}}$  on subsets of variables  $X_{\hat{\mathcal{V}}}$  is

$$\mathbb{P}^{\mathcal{G}}[X_{\hat{\mathcal{V}}}] = \langle \tau^{\mathcal{G}} \rangle_{[X_{\hat{\mathcal{V}}}|\emptyset]}.$$

This can be derived from a commutativity of contractions, which established an equivalence of contractions with sequences of consecutive contractions.

**Theorem 17 (Commutativity of Contractions)** *Let  $\tau^{\mathcal{G}}$  be a tensor network on a hypergraph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ . Let us now split the  $\mathcal{G}$  into two graphs  $\mathcal{G}_1 = (\mathcal{V}^1, \mathcal{E}_1)$  and  $\mathcal{G}_2 = (\mathcal{V}^2, \mathcal{E}_2)$ , such that  $\mathcal{E}_1 \dot{\cup} \mathcal{E}_2 = \mathcal{E}$ ,  $\mathcal{V}^1 \cup \mathcal{V}^2 = \mathcal{V}$  and all nodes in  $\mathcal{V}^2$  are contained in an hyperedge of  $\mathcal{E}_2$ . We then have for any  $\tilde{\mathcal{V}} \subset \mathcal{V}$*

$$\langle \tau^{\mathcal{G}} \rangle_{[X_{\tilde{\mathcal{V}}}] = \left\langle \tau^{\mathcal{G}_1} [X_{\mathcal{V}^1}] \cup \{ \langle \tau^{\mathcal{G}_2} \rangle_{[X_{\mathcal{V}^2 \cap (\mathcal{V}^1 \cup \tilde{\mathcal{V}})}]} \} \right\rangle_{[X_{\tilde{\mathcal{V}}}] .}$$

Further, the distribution of  $X_{\tilde{\mathcal{V}}}$  conditioned on  $X_{\bar{\mathcal{V}}}$ , where  $\tilde{\mathcal{V}}, \bar{\mathcal{V}}$  are disjoint subsets of  $\mathcal{V}$ , is

$$\mathbb{P}^{\mathcal{G}} [X_{\tilde{\mathcal{V}}} | X_{\bar{\mathcal{V}}}] = \langle \tau^{\mathcal{G}} \rangle_{[X_{\tilde{\mathcal{V}}} | X_{\bar{\mathcal{V}}}] .}$$

While we have directly defined Markov Networks as decomposed probability distributions, we now want to derive assumptions on a distribution assuring that such decompositions exist. As we will see, the sets of conditional independencies encoded by a hypergraph are captured by its separation properties, as we define next.

**Definition 18 (Separation of Hypergraph)** *A path in a hypergraph is a sequence of nodes  $v_k$  for  $k \in [d]$ , such that for any  $k \in [d - 1]$  we find a hyperedge  $e \in \mathcal{E}$  such that  $(v_k, v_{k+1}) \subset e$ . Given disjoint subsets  $A, B, C$  of nodes in a hypergraph  $\mathcal{G}$  we say that  $C$  separates  $A$  and  $B$  with respect to  $\mathcal{G}$ , when any path starting at a node in  $A$  and ending in a node in  $B$  contains a node in  $C$ .*

To characterize Markov Networks in terms of conditional independencies we need to further define the property of clique-capturing. This property of clique-capturing established a correspondence of hyperedges with maximal cliques in the more canonical graph-based definition of Markov Networks Koller and Friedman.

**Definition 19 (Clique-Capturing Hypergraph)** *We call a hypergraph  $\mathcal{G}$  clique-capturing, when each subset  $\tilde{\mathcal{V}} \subset \mathcal{V}$  is contained in a hyperedge, if for any  $a, b \in \tilde{\mathcal{V}}$  there is a hyperedge  $e \in \mathcal{E}$  with  $a, b \in \tilde{\mathcal{V}}$ .*

Let us now show a characterization of Markov Networks in terms of conditional independencies.

**Theorem 20 (Hammersley-Clifford)** *Given a clique-capturing hypergraph  $\mathcal{G}$ , the set of positive Markov Networks on the hypergraph coincides with the set of positive probability distributions, such that for each disjoint subsets of variables  $A, B, C$  we have  $X_A$  is independent of  $X_B$  conditioned on  $X_C$ , when  $C$  separates  $A$  and  $B$  in the hypergraph.*

Equivalently, Thm. 20 states that for any strictly positive joint distribution  $\mathbb{P}[X_V]$  whose conditional independencies are exactly those encoded by a clique-capturing hypergraph  $G = (V, E)$ , there exist non-negative activation cores  $\tau_e[X_e]$  such that

$$P[X_V] = \frac{1}{Z} \langle \{ \tau_e : e \in E(G) \} \rangle [X_V],$$

for a suitable normalizing constant  $Z > 0$ . Thus, the conditional-independence structure of  $\mathbb{P}$  determines a global tensor-network decomposition of its activation (and hence joint) tensor. We refer to this correspondence between independence structure and tensor-network factorization as the *independence mechanism*, in analogy to the computation mechanism provided by sufficient statistics in Section 3.1.

### 3.3 The Computation mechanism: Factorization in presense of Sufficient Statistics

A sufficient statistic is defined as a tensor, for which  $X_{[d]}$  and  $\mathbb{P}[X_{[d]}]$  are conditionally independent, when conditioned onto the tensor.

**Definition 21 (Sufficient Statistics)** *Let  $X_{[d]}$  be a list of by  $\mathbb{P}[X_{[d]}]$  jointly distributed random variables and  $\mathcal{S}(X_{[d]}, L)$  be a tensor. We say that  $\mathcal{S}$  is a sufficient statistic for  $\mathbb{P}$ , if for all  $y$  in the image of  $\mathcal{S}$  we have*

$$\mathbb{P}[X_{[d]} = x_{[d]} | \mathcal{S}(x_{[d]}) = y] = \begin{cases} \frac{1}{|\{x_{[d]} : \mathcal{S}(x_{[d]}) = y\}|} & \text{if } \mathcal{S}(x_{[d]}) = y \\ 0 & \text{else} \end{cases}.$$

When knowing the value  $\mathcal{S}x_{[d]}$  of the sufficient statistic at a given index  $x_{[d]}$ , we also know the probability  $\mathbb{P}[X_{[d]} = x_{[d]}]$ .

**Example 5 (Coin toss experiment)** *Let there be  $d$  boolean variables  $X_{[d]}$ . We have a coin toss family, if the sum is a sufficient statistic.*

*For the case  $d = 2$ : Consider two coin tosses  $X_1, X_2 \in \{0, 1\}$  ( $1$ =heads). With  $p \in [0, 1]$  being the probability of heads. Define the statistic*

$$S(X_1, X_2) = X_1 + X_2 \in \{0, 1, 2\}.$$

*Intuitively,  $S$  forgets order and keeps only the number of heads. The conditional law of the sequence given  $S$  is uniform over all sequences with that many heads:*

$$\mathbb{P}((X_1, X_2) = (x_1, x_2) | S = k) = \frac{1}{\binom{2}{k}} \quad \text{whenever } x_1 + x_2 = k.$$

*Thus, knowing  $S$  renders the detailed order uninformative about the distribution—this matches the idea of a sufficient statistic.*

**Theorem 22** *The tensor  $\mathcal{S}(X_{[d]}, L)$  is a sufficient statistic, i.e.  $X_{[d]}$  is independent of  $\mathbb{P}[X_{[d]}]$  conditioned on  $\mathcal{S}(X_{[d]}, L)$  if and only if there is a tensor  $\xi[Y_{[p]}]$  with*

$$\mathbb{P}[X_{[d]}] = \langle \beta^{\mathcal{S}}[Y_{[p]}, X_{[d]}], \xi[Y_{[p]}] \rangle_{[X_{[d]}]}.$$

The tensors in a decomposition of  $\xi[Y_{[p]}]$  are called *activation cores*. The tensors in a decomposition of  $\beta^{\mathcal{S}}[Y_{[p]}, X_{[d]}]$  are called *computation cores*.

**Example 6** *Unfair and dependent coin toss — Factorization as a Computation-Activation Network* Let  $X_1, X_2 \in \{0, 1\}$  and define the statistic  $S(X_1, X_2) = X_1 + X_2 \in \{0, 1, 2\}$ . The basis (computation) core for this statistic is

$$\beta_S[y, x_1, x_2] = \mathbf{1}\{y = x_1 + x_2\}, \quad y \in \{0, 1, 2\}.$$

For  $p \in (0, 1)$ , define a unary activation (a vector)  $\xi[Y]$  with components  $\xi[0] = (1 - p)^2$ ,  $\xi[1] = (1 - p)p$ ,  $\xi[2] = p^2$ . The joint distribution factors as

$$\mathbb{P}[X_1, X_2] = \frac{1}{Z} \langle \beta_S[Y, X_1, X_2], \xi[Y] \rangle_Y$$

with

$$\begin{aligned} Z &:= \langle \langle \beta_S[Y, X_1, X_2], \xi[Y] \rangle_Y \rangle_{[\emptyset]} \\ &= \sum_{y=0}^2 \xi[y] \sum_{x_1=0}^1 \sum_{x_2=0}^1 \beta_S[y, x_1, x_2] = 1\xi[0] + 2\xi[1] + 1\xi[2] \\ &= (1 - p)^2 + 2(1 - p)p + p^2 = (p + (1 - p))^2 = 1 \end{aligned}$$

Equivalently, in coordinates,

$$\mathbb{P}[x_1, x_2] = \frac{1}{Z} \sum_{y=0}^2 \beta_S[y, x_1, x_2] \xi[y] = \xi(S(x_1, x_2)).$$

Since  $S(0, 0) = 0$ ,  $S(0, 1) = S(1, 0) = 1$ , and  $S(1, 1) = 2$ , the induced law of the statistic is

$$\mathbb{P}[S = 0] = (1 - p)^2, \quad \mathbb{P}[S = 1] = 2p(1 - p), \quad \mathbb{P}[S = 2] = p^2,$$

reflecting that there is only one configuration for  $y \in \{0, 2\}$  and 2 configurations for  $y = 1$ .

### 3.4 Exponential families in case of elementary activation tensors

Further motivation: The Pitman-Koopman-Darmois theorem, stating that finite sufficient statistics (for the likelihood of a sequence of independent data) are only possible in exponential families, given that the support is constant.

We now restrict the activation cores to specific elementary tensors, which correspond with further assumptions on the dependence of  $\mathbb{P}$  and  $\mathcal{S}$  made by exponential families.

**Definition 23 (Exponential Family)** *from Thm. ??* Given any base measure  $\nu$  and a sufficient statistic  $\mathcal{S}$  we enumerate for each coordinate  $l \in [p]$  the image  $\text{im}(s_l)$  by a variable  $Y_l$  taking values in  $[\text{im}(s_l)]$  (see for more details on this scheme Chapter ??), given an interpretation map

$$I_l : [\text{im}(s_l)] \rightarrow \text{im}(s_l).$$

For any canonical parameter vector  $\theta[L] \in \mathbb{R}^p$  we build the activation cores  $\alpha^{l, \theta}[Y_l]$  for each coordinate  $y_l \in [\text{im}(s_l)]$  by

$$\alpha^{l, \theta}[Y_l = y_l] = \exp[\theta[L = l] \cdot I_l(y_l)]$$

and have

$$\mathbb{P}^{(\mathcal{S}, \theta, \nu)} [X_{[d]}] = \left\langle \{\nu [X_{[d]}]\} \cup \{\beta^{sl} [Y_l, X_{[d]}] : l \in [p]\} \cup \{\alpha^{l, \theta} [Y_l] : l \in [p]\} \right\rangle_{[X_{[d]} | \emptyset]} .$$

We then call the set

$$\Gamma^{\mathcal{S}, \nu} = \{\mathbb{P}^{(\mathcal{S}, \theta, \nu)} : \theta [L] \in \mathbb{R}^p\}$$

the exponential family with respect to the statistic  $\mathcal{S}$  and the base measure  $\nu$ .

**Example 7** *Joint distributions of two Booleans, sufficient statistic by head count (coin toss interpretation) and exponential family in case of independence* In general, joint distribution of two Boolean variables  $X_1, X_2$  are  $2 \times 2$  matrices of non-negative coordinates summing to 1:

$$\mathbb{P}[X_1, X_2] = \begin{bmatrix} p_{0,0} & p_{0,1} \\ p_{1,0} & p_{1,1} \end{bmatrix}$$

In the following example, we will assume at different points that  $X_1, X_2$  have a sufficient statistic, are independent and they have positive distributions. By the normalization constraint,  $p_{1,1}$  is determined from  $p_{0,0}, p_{0,1}$  and  $p_{1,0}$ , which leaves us with three free parameters.

$$\mathbb{P}[X_1, X_2] = \begin{bmatrix} p_{0,0} & p_{0,1} \\ p_{1,0} & 1 - (p_{0,0} + p_{0,1} + p_{1,0}) \end{bmatrix}$$

Let us now restrict to those distributions, which have the sum  $X_1 + X_2$  as a sufficient statistic. They need to satisfy  $p_{0,1} = p_{1,0}$  (since in that cases the statistic is 1 and the definition of sufficiency is that the distribution conditioned on the statistic is uniform), leaving us with two free parameters.

$$\mathbb{P}[X_1, X_2] = \begin{bmatrix} p_{0,0} & p_{0,1} \\ p_{0,1} & 1 - p_{0,0} - 2p_{0,1} \end{bmatrix}$$

This symmetry also implies, that the distributions are identically distributed with  $\mathbb{I}_2 = (1, 1)^\top$ :

$$\mathbb{P}[X_1] = \langle \mathbb{P}[X_1, X_2], \mathbb{I}[X_2] \rangle [X_1] = \mathbb{P}[X_1, X_2] \mathbb{I}_2 = \mathbb{P}[X_2, X_1] \mathbb{I}_2 = \langle \mathbb{P}[X_1, X_2], \mathbb{I}[X_1] \rangle [X_2] = \mathbb{P}[X_2] .$$

Restricting further to those, where  $X_1$  and  $X_2$  are independent and the distribution is everywhere supported, brings us to the rank one formulation of the distribution

$$\mathbb{P}[X_1, X_2] = \begin{bmatrix} \mathbb{P}[X_1 = 0] \mathbb{P}[X_2 = 0] & \mathbb{P}[X_1 = 0] \mathbb{P}[X_2 = 1] \\ \mathbb{P}[X_1 = 1] \mathbb{P}[X_2 = 0] & \mathbb{P}[X_1 = 1] \mathbb{P}[X_2 = 1] \end{bmatrix} = \mathbb{P}[X_1] \mathbb{P}[X_2]^\top = \mathbb{P}[X_1] \mathbb{P}[X_1]^\top .$$

In terms of an exponential family with the head count as a sufficient statistic, we parametrize the distribution by the canonical parameter  $\theta \in \mathbb{R}$  as

$$\mathbb{P}[X_1] = \frac{1}{1 + \exp[\theta]} \begin{bmatrix} 1 \\ \exp[\theta] \end{bmatrix}$$

Note, that with this parametrization the probabilities for head and tail automatically have the form  $p, (1 - p)$ .

$$\mathbb{P}[X_1, X_2] = \frac{1}{(1 + \exp[\theta])^2} \begin{bmatrix} 1 \\ \exp[\theta] \end{bmatrix} \begin{bmatrix} 1 & \exp[\theta] \end{bmatrix}$$

We can interpret this distribution as two independent coin tosses with outcome  $X_1$  and  $X_2$  and head probability

$$\mathbb{P}[X_1 = 1] = \mathbb{P}[X_2 = 1] = \frac{\exp[\theta]}{1 + \exp[\theta]}$$

which is the sigmoid of  $\theta$  and inverted by the logit

$$\theta = \ln \left[ \frac{\mathbb{P}[X_1 = 1]}{1 - \mathbb{P}[X_1 = 1]} \right].$$

Consistent with the above parametrization, we have a uniform distribution of  $X_1$  and  $X_2$  in the fair coin toss case  $\mathbb{P}[X_1 = 1] = 0.5$ , where  $\theta = 0$ .

As a Computation-Activation Network we can represent any distribution  $\mathbb{P}[X_1, X_2]$  with the head count  $+$  as sufficient statistic by

$$\mathbb{P}[X_1, X_2] = \langle \beta^+[Y, X_1, X_2], \xi[Y] \rangle_{[X_1, X_2]|\emptyset},$$

such that

$$\begin{aligned} \mathbb{P}[X_1 = x_1, X_2 = x_2] &= \frac{1}{Z} \langle \beta^+[Y, X_1, X_2], \xi[Y] \rangle_{[X_1=x_1, X_2=x_2]} \\ &= \frac{1}{Z} \sum_{y=0}^2 \beta^+[Y = y, X_1 = x_1, X_2 = x_2] \xi[y] = \frac{1}{Z} \xi[Y = x_1 + x_2], \end{aligned}$$

where the normalization constant  $Z$  cancels out any multiplicative constant  $\lambda \in \mathbb{R} \setminus \{0\}$  in  $\xi$  and the equation above implies

$$\xi[Y] = \begin{bmatrix} p_{0,0}\lambda \\ p_{0,1}\lambda \\ p_{1,1}\lambda \end{bmatrix}.$$

We choose  $\lambda = 1/p_{0,0} = (1 + \exp[\theta])^2$  in the following. Among these distribution, the exponential family with the head count statistic is then parametrized by activation tensors

$$\xi[Y] = \begin{bmatrix} 1 \\ p_{0,1}/p_{0,0} \\ p_{1,1}/p_{0,0} \end{bmatrix} = \begin{bmatrix} 1 \\ \exp[\theta] \\ \exp[2\theta] \end{bmatrix},$$

since  $p_{0,1} = \mathbb{P}[X_1 = 0] \cdot \mathbb{P}[X_1 = 1] = (1 + \exp[\theta])^{-1} \cdot \exp[\theta] (1 + \exp[\theta])^{-1}$  and  $p_{1,1} = (\exp[\theta] (1 + \exp[\theta])^{-1})^2$ .

### 3.5 Contractions to compute marginal probabilities

The most central inference operation on probability distributions is the computation of marginals, which is just the contraction leaving only the marginalized variable open. Conditional probabilities are marginal probabilities of the distribution with added boolean tensors marking the evidence (e.g. one-hot encodings to states of some variables). With this formalism we can answer queries like "What is the probability of booking this account when having observed this feature on the invoice?".

Having described how independence structures induce sparse tensor-network factorizations of the activation (and hence joint) tensor via the Hammersley-Clifford theorem (Thm. 20) in the previous section, we now turn to inference. In this section we explain how standard inference queries, namely marginals and conditional probabilities, are realized as tensor contractions of the resulting decomposed network.

In the graphical-model CAN with identity statistic, the joint distribution is represented by the activation tensor  $\xi[X_V] = \mathbb{P}[X_V]$  on the variable set  $V$ . For any subset  $A \subseteq V$ , the marginal distribution on  $X_A$  is obtained by contracting all non-query legs with trivial tensors  $I[X_v]$ :

$$\mathbb{P}[X_A] = \langle \mathbb{P}[X_V], \{I[X_v] : v \in V \setminus A\} \rangle [X_A].$$

In words, we close all legs corresponding to variables outside  $A$  with identity tensors and read off the resulting tensor on the remaining open legs  $X_A$ . In the Markov/activation network representation of Section 3.3, computing a marginal thus corresponds to closing all non-query nodes and edges and evaluating the resulting contracted network on the query variables.

We can interpret conditionals as normalized evidence slices. Let  $A, C \subseteq V$  and fix evidence  $X_C = x_C$ . Using the one-hot encodings  $\epsilon_{x_c}[X_c]$  from Def. 2, we define the unnormalized slice

$$\tilde{\mathbb{P}}[X_A \mid X_C = x_C] := \langle \mathbb{P}[X_V], \{\epsilon_{x_c}[X_c] : c \in C\}, \{I[X_v] : v \in V \setminus (A \cup C)\} \rangle [X_A].$$

The corresponding conditional distribution is obtained by normalization,

$$\mathbb{P}[X_A \mid X_C = x_C] = \frac{\tilde{\mathbb{P}}[X_A \mid X_C = x_C]}{\tilde{\mathbb{P}}[X_A \mid X_C = x_C][\emptyset]} = \frac{\tilde{\mathbb{P}}[X_A \mid X_C = x_C]}{\sum_{x_A} \tilde{\mathbb{P}}[X_A = x_A \mid X_C = x_C]}. \quad (5)$$

Thus conditional probabilities are obtained by contracting the activation/joint tensor with one-hot evidence tensors and renormalizing the resulting slice. In a Computation-Activation Network, inference with evidence therefore corresponds to contracting the activation network with one-hot tensors encoding observed variables and reading out the resulting activation on the query legs.

[Accounting example here?](#)

## 4 Logical representation: Propositional formulas through hard activation

A tensor-based representation of propositional logic is developed by encoding boolean variables into vectors, defining formulas as boolean tensors, and showing how logical connectives and normal forms can be expressed as tensor contractions.

Here propositional logic is based on propositional formulas. Define

- a *propositional formula* as a boolean-valued tensor

$$f [X_{[d]}] : \bigtimes_{k \in [d]} [2] \rightarrow \{0, 1\} \subset \mathbb{R},$$

- a *model* of a propositional formula as a state  $x_{[d]} \in \bigtimes_{k \in [d]} [2]$ , which fulfills  $f [X_{[d]} = x_{[d]}] = 1$ , where we associate  $\text{True} \leftrightarrow 1$  and  $\text{False} \leftrightarrow 0$ ,
- and the propositional formula to be *satisfiable*, if there is a model.

**Example 8** The propositional formula defined for  $d = 3$  and  $x_{[3]} = (x_1, x_2, x_3) \in \bigtimes_{k \in [d]} [2] = \{0, 1\} \times \{0, 1\} \times \{0, 1\}$  by

$$f [X_{[d]} = x_{[3]}] = x_1 \wedge (x_2 \vee x_3)$$

is satisfiable, since  $f [X_{[d]} = (1, 1, 1)] = 1$ ,  $f [X_{[d]} = (1, 0, 1)] = 1$ , and  $f [X_{[d]} = (1, 1, 0)] = 1$  and therefore  $x = (1, 1, 1)$ ,  $x = (1, 0, 1)$ , and  $x = (1, 1, 0)$  are models of  $f [X_{[d]}]$ .

Representing booleans by elements in  $\{0, 1\}$  leads to the problem, that negation is an affine transformation and can not be represented by multilinear tensors (Goessmann, 2025, Section 4.1.1). Therefore, instead of using this *coordinate calculus* an approach based on *basis calculus* is employed, which is explained in this section.

This representation of propositional formulas with respect to basis encoding also leads to Computation-Activation Networks, which were also used to describe probability distributions in the last section. In this way the soft and hard logic can be combined in one framework.

#### 4.1 Propositional Semantics by Boolean Tensors

To be able to express different kinds of connectives and finally any propositional formula by multi-linear tensors, booleans are encoded by one-hot encodings as defined in Definition 2.

There are different ways to represent propositional formulas based on tensor decompositions and basis encoding representation tensors.

**CP decomposition** Since the tensor  $f [X_{[d]}]$  is equal to one at index  $x_{[d]}$  if and only if  $x_{[d]}$  is a model of  $f$ , i.e. fulfills the formula, A propositional formula can be written as the sum over the one-hot encodings of its models.

$$\begin{array}{c} \boxed{f} \\ \begin{array}{c} x_0 \quad | \quad x_1 \quad | \quad \dots \quad | \quad x_{d-1} \end{array} \end{array} = \sum_{\substack{x_0, \dots, x_{d-1} \in \bigtimes_{k \in [d]} [2] \\ f(x_0, \dots, x_{d-1}) = 1}} \begin{array}{c} \boxed{\epsilon_{x_0}} \\ \downarrow^{x_0} \end{array} \dots \begin{array}{c} \boxed{\epsilon_{x_{d-1}}} \\ \downarrow^{x_{d-1}} \end{array}$$

This decomposition corresponds to the CP decomposition of a tensor.

**Example 9** For the formula described in Example 8, we have

$$f [X_{[3]}] = (\epsilon_1 [X_1] \otimes \epsilon_1 [X_2] \otimes \epsilon_1 [X_3]) + (\epsilon_1 [X_1] \otimes \epsilon_0 [X_2] \otimes \epsilon_1 [X_3]) + (\epsilon_1 [X_1] \otimes \epsilon_1 [X_2] \otimes \epsilon_0 [X_3]),$$

where  $\epsilon_1[Y] = (0, 1)^T$  and  $\epsilon_0[Y] = (1, 0)^T$ . Then for the model  $x_{[3]} = (1, 0, 1)$  it holds

$$\begin{aligned} f[X_{[3]} = x_{[3]}] &= (\epsilon_1[X_1 = 1] \otimes \epsilon_1[X_2 = 0] \otimes \epsilon_1[X_3 = 1]) + (\epsilon_1[X_1 = 1] \otimes \epsilon_0[X_2 = 0] \otimes \epsilon_1[X_3 = 1]) \\ &\quad + (\epsilon_1[X_1 = 1] \otimes \epsilon_1[X_2 = 0] \otimes \epsilon_0[X_3 = 1]) \\ &= 1 \cdot 0 \cdot 1 + 1 \cdot 1 \cdot 1 + 1 \cdot 0 \cdot 0 = 1. \end{aligned}$$

**Basis encoding** Propositional formulas  $f$  can be expressed in terms of a tensor describing the mapping and its negation by

$$\beta^f[Y_f = y, X_{[d]} = x_{[d]}] = \begin{cases} 1, & f[X_{[d]} = x_{[d]}] = y \\ 0, & \text{else} \end{cases}. \quad (6)$$

This basis encoding  $\beta^f[Y_f, X_{[d]}] \in \{0, 1\}^{2 \times 2^d}$  then has the form

$$\beta^f[Y_f, X_{[d]}] = f[X_{[d]}] \otimes \epsilon_1[Y_f] + \neg f[X_{[d]}] \otimes \epsilon_0[Y_f]. \quad (7)$$

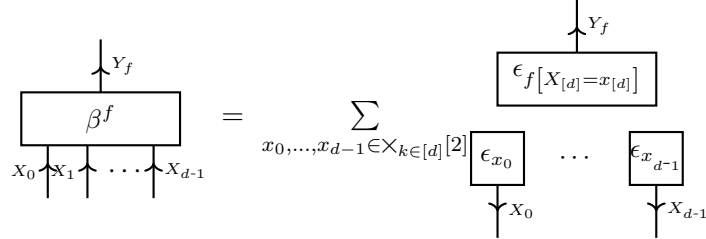
Then the propositional formula and its negation can be represented by that tensor by

$$f[X_{[d]}] = \left\langle \beta^f[Y_f, X_{[d]}], \epsilon_1[Y_f] \right\rangle_{[X_{[d]}]}$$

and

$$\neg f[X_{[d]}] = \left\langle \beta^f[Y_f, X_{[d]}], \epsilon_0[Y_f] \right\rangle_{[X_{[d]}]}.$$

As in (Goessmann, 2025, Section 4.2.2), in graphical notation the basis encoding has the following form.



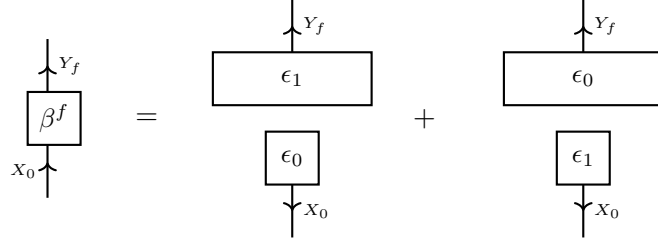
**Example 10** For the formula  $f[X_{[1]} = x_1] = \neg x_1$ , i.e.  $f[X_{[1]}] = (1, 0)^T$ , we have with (6)

$$\begin{aligned} \beta^f[Y_f, X_{[1]}] &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{matrix} \leftarrow Y_f = 0 \\ \leftarrow Y_f = 1 \end{matrix} \\ &\quad \begin{matrix} \uparrow \\ X_1 = 0 \end{matrix} \begin{matrix} \uparrow \\ X_1 = 1 \end{matrix} \end{aligned}$$

with (7) we have

$$\beta^f[Y_f, X_{[1]}] = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

and in graphical notation we get



**Tensor network decomposition** If the propositional formula allows for a decomposition into smaller propositional formulas, the representation can be reduced by employing a tensor network structure as described in the next subsection.

**Model counts** At each index the propositional formula either carry a 1 or 0 encoding if the indexed value is true or false. This implies, that contracting the tensor with trivial 1-tensors applied to each open leg leaving only one leg open results in a vector counting the number of models of the formula for each possible index of the open leg. Furthermore, contracting the whole tensor yields the number of models of the given formula. This is consistent with a probabilistic interpretation discussed in later.

## 4.2 Propositional Syntax leading to Tensor-Network Decompositions

**Definition 24** Define an  $r$ -ary connective as a function  $\circ : \times_{\ell \in [r]} [2] \rightarrow [2]$ .

**Example 11** As in (Goessmann, 2025, Example 4.9) the following connectives are of interest.

- *negation*  $\neg : [2] \rightarrow [2]$  by the vector

$$\neg[Y_f] = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- *conjunctions*  $\wedge : [2] \times [2] \rightarrow [2]$

$$\wedge[Y_f, Y_h] = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

- *disjunctions*  $\vee : [2] \times [2] \rightarrow [2]$

$$\vee[Y_f, Y_h] = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

- *exact disjunction*  $\oplus : [2] \times [2] \rightarrow [2]$

$$\oplus[Y_f, Y_h] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

- *implications*  $\Rightarrow : [2] \times [2] \rightarrow [2]$

$$\Rightarrow[Y_f, Y_h] = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

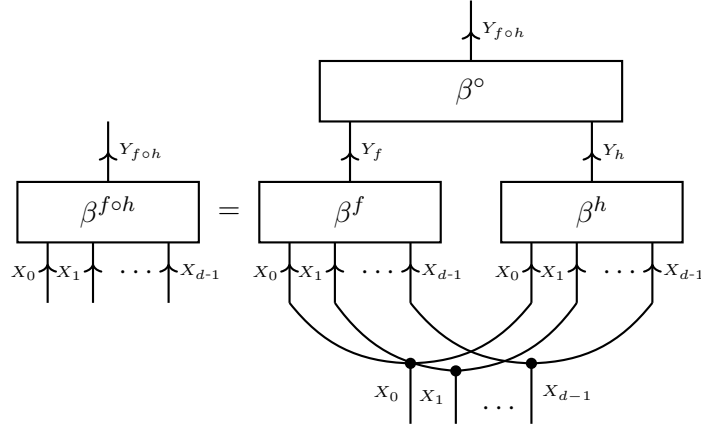
- *bimplication*  $\Leftrightarrow: [2] \times [2] \rightarrow [2]$

$$\Leftrightarrow [Y_f, Y_h] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

If  $f [X_{[d]}]$  is defined as the  $r$ -ary composition of propositional formulas  $f_\ell, \ell \in [r]$ , i.e.

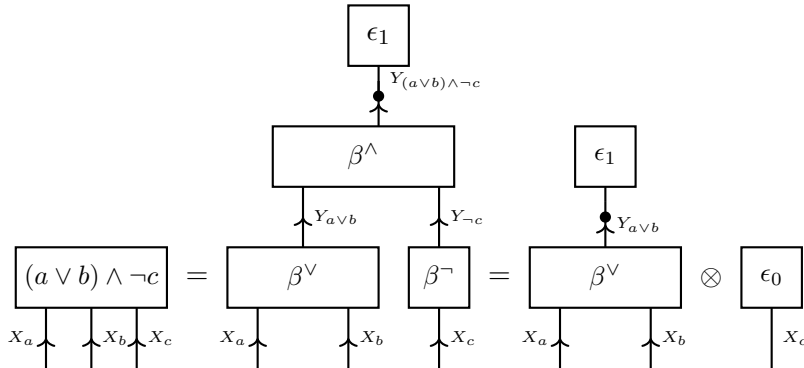
$$f [X_{[d]} = x_{[d]}] = \circ (f_0 [X_{[d]} = x_{[d]}], \dots, f_{r-1} [X_{[d]} = x_{[d]}]),$$

then the basis encoding can be written as the contraction of the basis encoding of the  $r$ -ary composition and the basis encodings of the individual propositional formulas  $f_\ell, \ell \in [r]$ . For the composition of two propositional formulas  $f [X_{[d]}]$  and  $h [X_{[d]}]$  the composition by some binary connective is pictured below.



The figure can be found in (Goessmann, 2025, Figure 4.2 b)), where hyperedges made up by more than two nodes represent the unaltered connection between nodes.

**Example 12** (see the notebook: <https://colab.research.google.com/drive/1p2wp61fFMu0otnfFhKoNsusp=sharing>) For the example described in (Goessmann, 2025, Figure 4.2) with the propositional formula  $f [X_{[3]} = (a, b, c)] = (a \vee b) \vee \neg c$ , we can write the formula in terms of a Computation-Activation Network with activation tensor  $\epsilon_1$  and computation network decomposed by the basis encodings. First, it is written with one activation vector. Second, we see that it can also be interpreted with multiple features.



### 4.3 Contractions to decide entailment

As in (Goessmann, 2025, Definition 5.1), we define the entailment of two propositional formulas as follows.

**Definition 25 (Entailment of propositional formulas)** *Given two propositional formulas  $\mathcal{KB}$  and  $f$  we say that  $\mathcal{KB}$  entails  $f$ , denoted by  $\mathcal{KB} \models f$ , if any model of  $\mathcal{KB}$  is also a model of  $f$ , that is*

$$\forall x_{[d]} \in \prod_{k \in [d]} [2] : (\mathcal{KB} [X_{[d]} = x_{[d]}] = 1) \Rightarrow (f [X_{[d]} = x_{[d]}] = 1).$$

If  $\mathcal{KB} \models \neg f$  holds, we say that  $\mathcal{KB}$  contradicts  $f$ .

This property of interest, since  $\mathcal{KB}$  can be chosen as a knowledge base, which can be interpreted in different ways: as the support of probability distributions,

This is the central operation of "logical inference", i.e. deduce true statements from known statements. One can in some cases understand this as "making the knowledge base more accessible": Adding deduced statements to a knowledge base does not change the knowledge base as a tensor, but one can interpret it in an easier way.

**Example 13** *Attempt to match the above Sudoku example with our notation of boolean variables and the entailment formalism* We index the rows and the columns by tuples  $(r0, r1)$  and  $(c0, c1)$ , where  $r0, r1, c0, c1 \in [3]$ . The first index indicates the block and the second counts the row or column inside that block. For each  $r0, r1, c0, c1 \in [3]$  and  $n \in [9]$  we then define an atomic variable  $X_{r0, r1, c0, c1, n} \in \{0, 1\}$  indicating whether in the row  $(r0, r1)$  and column  $(c0, c1)$  the number  $n$  is written. The Sudoku rules then amount to the formula

$$\begin{aligned} \mathcal{KB} = & \left( \bigwedge_{r0, r1, c0, c1 \in [3]} \left( \bigoplus_{n \in [9]} X_{r0, r1, c0, c1, n} \right) \right) \wedge \left( \bigwedge_{r0, r1 \in [3], n \in [9]} \left( \bigoplus_{c0, c1 \in [3]} X_{r0, r1, c0, c1, n} \right) \right) \wedge \\ & \left( \bigwedge_{c0, c1 \in [3], n \in [9]} \left( \bigoplus_{r0, r1 \in [3]} X_{r0, r1, c0, c1, n} \right) \right) \wedge \left( \bigwedge_{r0, c0 \in [3], n \in [9]} \left( \bigoplus_{r1, c1 \in [3]} X_{r0, r1, c0, c1, n} \right) \right), \end{aligned}$$

where  $\bigoplus$  is the 9-ary exclusive or connective (that is 1 if and only if exactly one of the arguments is 1). The four outer brackets in  $\mathcal{KB}$  mark the constraints, that at each position exactly one number is assigned, further that in each row each number is assigned once, and similar for the columns and the squares of the board. When solving a specific Sudoku instance, one typically knows from an initial board assignment  $E$  a collection of atomic variables, which hold, and needs to find further atomic variables, which are entailed. This means, we need to decide for each  $(r0, r1, c0, c1, n) \notin E$  whether the Sudoku rules and the initial board imply that the atomic variable  $X_{r0, r1, c0, c1, n}$  (i.e. assignment to the board) is true

$$\left( \mathcal{KB} \wedge \bigwedge_{(r0, r1, c0, c1, n) \in E} X_{r0, r1, c0, c1, n} \right) \models X_{r0, r1, c0, c1, n}$$

or false

$$\left( \mathcal{KB} \wedge \bigwedge_{(r_0, r_1, c_0, c_1, n) \in E} X_{r_0, r_1, c_0, c_1, n} \right) \models \neg X_{r_0, r_1, c_0, c_1, n} . \quad (8)$$

In other words, for each assignment to the board, that fulfills the Sudoku rules and the initial board, do we write the number  $n$  in row  $(r_0, r_1)$  and column  $(c_0, c_1)$ ? If and only if the Sudoku has a unique solution given the initial board assignment  $E$ , exactly one of these entailment statements holds for each  $(r_0, r_1, c_0, c_1, n) \notin E$ . Deciding which is equivalent to solving of the Sudoku.

In the tensor network representation, these entailments can be decided by contracting the whole representing tensor with the statement, that needs to be checked.

**Theorem 26 (Entailment in propositional logic (Goessmann, 2025, Theorem 5.2))**  
For two propositional formulas  $\mathcal{KB}$ ,  $f$  we have

$$\mathcal{KB} \models f \iff \langle \mathcal{KB}, \neg f \rangle_{[\emptyset]} = 0 .$$

Since contracting the whole tensor is often infeasible and for instance for the Sudoku example would correspond to solving the whole problem, local contractions can be considered to decide in some cases. Here a local contraction describes the calculation of contractions along few closely connected legs in the tensor network. Now, if the local contraction of any legs leads to a zero-tensor in the network decomposition, the whole contraction amounts to zero, and the knowledge base entails  $f$ .

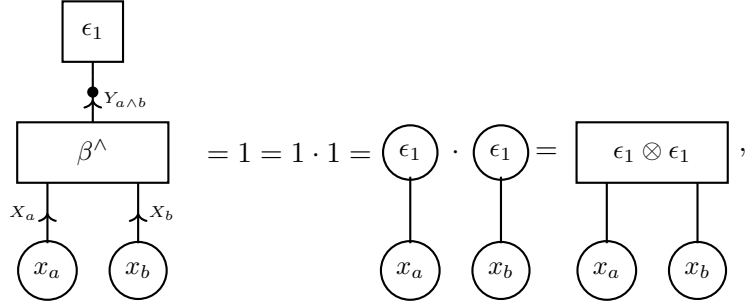
**Example 14 Sudoku continued** For the Sudoku example, a local contraction of the left- and right-hand-side of (8) means that only direct connections via the basic Sudoku rules and the known board are checked. If the local contraction amounts to zero, the direct Sudoku rules already imply, that  $X_{r_0, r_1, c_0, c_1, n}$  must be true/is the only possible choice. In most cases the local contraction will not lead to 0. Then no statement can be made.

#### 4.4 Representation of Knowledge Bases

When having multiple formulas: Represent with multiple  $\epsilon_1$  activation cores (Goessmann, 2025, Theorem 14.36).

$$\langle \epsilon_1[Y], \beta^\wedge[Y, X_{[d]}] \rangle_{[X_{[d]}]} = \bigotimes_{k \in [d]} \epsilon_1[X_k]$$

Noting that for example for  $x_a = x_b = \epsilon_1 = [0, 1]^\top$



while for all other vectors  $x_a, x_b$ , all parts of the equations amount to 0. This yields that a knowledge base consisting of multiple formulas connected by a  $\wedge$  has an efficient representation by decomposing the the tensor into its individual formulas.

## 5 Hybrid Logic Networks

Logical and probabilistic reasoning can be unified in a single tensor-network architecture called a Hybrid Logic Network (HLN).

In Markov logic networks, each state gets assigned a real-valued probability based on the basis encoding of a boolean statistic as a computation network and assigned probabilities of these features in terms of a positive elementary activation network.

**Definition 27 (Markov Logic Network)** *A Markov Logic Networks is an element of and exponential family  $\Gamma^{\mathcal{F}, \mathbb{I}}$  with sufficient statistic defined coordinatewise by propositional formulas  $f_\ell$*

$$\mathcal{F} : \prod_{k \in [d]} [2] \rightarrow \prod_{l \in [p]} [2] \subset \mathbb{R}^p, \quad x_{[d]} \mapsto (f_\ell[X_{[d]} = x_{[d]}])_{\ell \in [p]}.$$

That means, they have the form

$$\mathbb{P}^{(\mathcal{F}, \theta, \mathbb{I})}[X_{[d]}] = \left\langle \beta^{\mathcal{F}}[Y_{[p]}, X_{[d]}], \alpha^\theta[Y_{[d]}] \right\rangle_{[X_{[d]} | \emptyset]}$$

with  $\alpha^\theta[Y_{[d]}] = \bigotimes_{l \in [p]} \alpha^{\ell, \theta}[Y_\ell]$  with

$$\alpha^{l, \theta}[Y_l = y_l] = \exp[\theta[L = l] \cdot I_l(y_l)]$$

where  $\theta[L] \in \mathbb{R}^p$ ,  $I$  is an interpretation map. The architecture is visualized in Figure 5.

In Hard Logic Networks, on the other hand, states only get assigned true or false values 0/1. Basis encodings of propositional formulas can be interpreted as a CAN, where the basis encodings build the computation network and the boolean elementary activation network decides, which outcomes of the statistic are possible/ get assigned a true value.

**Definition 28 (Hard Logic Network)** *Given a boolean statistic  $\mathcal{F} : \prod_{k \in [d]} [2] \rightarrow [2]^p$ , a subset  $A \subset [p]$  and a tuple  $y_A \in \prod_{l \in A} [2]$  the Hard Logic Network is the distribution*

$$\mathbb{P}^{\mathcal{F}, (A, y_A)}[X_{[d]}] = \left\langle \beta^{\mathcal{F}}[Y_{[p]}, X_{[d]}], \kappa^{(A, y_A)}[Y_{[p]}] \right\rangle_{[X_{[d]} | \emptyset]}$$

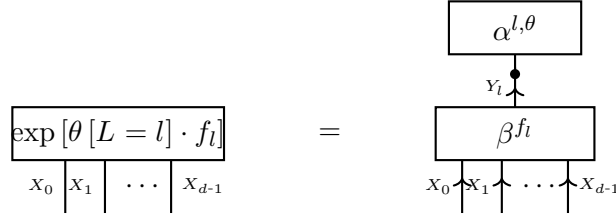


Figure 5: Factor of a Markov Logic Network to a formula  $f_l$ , represented as the contraction of a computation core  $\beta^{f_l}$  and an activation core  $\alpha^{l, \theta}$ . While the computation core  $\beta^{f_l}$  prepares based on basis calculus a categorical variable representing the value of the statistic formula  $f_l$  dependent on assignments to the distributed variables, the activation core multiplies an exponential weight to coordinates satisfying the formula.

where  $\kappa^{(A, y_A)} [Y_{[p]}] = \bigotimes_{l \in [p]} \kappa^{l, (A, y_A)} [Y_l]$  and for  $l \in [p]$

$$\kappa^{l, (A, y_A)} [Y_l] = \begin{cases} \epsilon_{y_l} [Y_l] & \text{if } l \in A \\ \mathbb{I} [Y_l] & \text{if } l \notin A \end{cases},$$

provided that the normalization exists.

Note that this distribution is uniform on its support. Combining the Hard Logic Network and the Markov Logic Network leads to the representation of a probability density only over states, that fulfill an imposed hard logic, where the elementary activation tensors are allowed to take binary or real values.

**Definition 29 (Hybrid Logic Network)** Given a boolean statistic  $\mathcal{F}$  we call any element of  $\Lambda^{\mathcal{F}, \text{EL}}$  a Hybrid Logic Network (HLN). The extended canonical parameter set to  $\mathcal{F}$  is the set

$$\mathcal{P}_p := \{(A, y_A) : A \subset [p], y_A \in \bigtimes_{l \in A} [2]\} \times \mathbb{R}^p.$$

To each Hybrid Logic Network  $\mathbb{P}^{\mathcal{F}, (A, y_A, \theta)} [X_{[d]}]$  we find a tuple  $(A, y_A, \theta)$  consistent of a subset  $A \subset [p]$ , a tuple  $y_A \in \bigtimes_{l \in A} [2]$  and  $\theta [L] \in \mathbb{R}^p$  such that

$$\mathbb{P}^{\mathcal{F}, (A, y_A, \theta)} [X_{[d]}] = \left\langle \beta^{\mathcal{F}} [Y_{[p]}, X_{[d]}], \xi^{(A, y_A, \theta)} [Y_{[p]}] \right\rangle_{[X_{[d]} | \emptyset]}$$

where the activation core is

$$\xi^{(A, y_A, \theta)} [Y_{[p]}] = \left\langle \alpha^{\theta} [Y_{[p]}], \kappa^{(A, y_A)} [Y_{[p]}] \right\rangle_{[Y_{[p]}]}.$$

The architecture is demonstrated on an example in accounting logic.

**Example 15** *Hybrid Logic Network for Accounting Logic*

Let us consider a system of three variables  $A_1$  Account 1 is booked,  $A_2$  Account 2 is booked,  $F$  a feature on an invoice. We respect two rules

- *Exactly one account must be booked.*
- *If feature  $F$  is present on the invoice, the account  $A_1$  is typically booked.*

We formalize this with the statistic

$$\mathcal{F} = (X_{A_1} \oplus X_{A_2}, X_F \Rightarrow X_{A_1}).$$

While the first formula is a hard feature, the second is soft since prone to exceptions. We parameterize the first output of the statistic with the hard parameters by setting the set of indices to be initialized with hard logic  $A = \{0\}$  and the corresponding initialization  $y_0 = 1$  meaning, that the first output of the statistic has to be true for the input to have positive probability. Then "hard logic activation tensor", should be indifferent to the second part of the statistic, and only impose rules on the first part, leading to

$$\kappa^{(A, y_A)}[Y_0, Y_1] = \epsilon_{y_0}[Y_0] \otimes \mathbb{I}[Y_1] = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

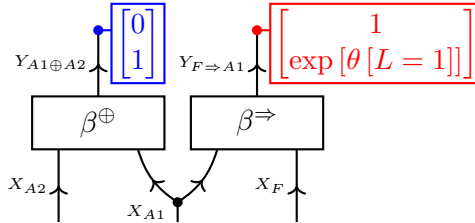
Since the first feature is hard, the "soft logic activation tensor" should be invariant under the first coordinate of the canonical parameter and we set  $\theta[L=0] = 0$ . We choose the soft parameters as  $\theta[L] = [0, \theta[L=1]]^\top$  to achieve

$$\alpha^\theta[Y_0, Y_1] = \alpha^{0,0}[Y_0] \otimes \alpha^{1, \theta[L=1]}[Y_1] = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \exp[\theta[L=1]] \end{bmatrix}.$$

The activation tensor of the hybrid network then has the form

$$\xi^{(A, y_A, \theta)}[Y_0, Y_1] = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \exp[\theta[L=1]] \end{bmatrix}.$$

We get a tensor network representation of the Hybrid Logic Network representing the toy accounting example, before normalization to a distribution



The resulting Hybrid Logic Network is a tensor  $\mathbb{P}^{\mathcal{F}, (A, y_A, \theta)}[X_{A_1}, X_{A_2}, X_F]$  of order 3. With  $Y_{F \Rightarrow A_1} = 1$  for  $F = 0$  and any  $A_1$  it has the coordinates

$$\begin{aligned} & \mathbb{P}^{\mathcal{F}, (A, y_A, \theta)}[X_{A_1}, X_{A_2}, X_F = 0] \\ &= \left\langle \beta^\oplus[Y_{A_1 \oplus A_2}, X_{A_2}, X_{A_1}] \otimes \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} [Y_{F \Rightarrow A_1}, X_{A_1}], \begin{bmatrix} 0 \\ 1 \end{bmatrix} [Y_{A_1 \oplus A_2}] \otimes \begin{bmatrix} 1 \\ \exp[\theta[L=1]] \end{bmatrix} [Y_{F \Rightarrow A_1}] \right\rangle \end{aligned}$$

where contraction along the  $Y$ -variables leads to

$$\begin{aligned}\mathbb{P}^{\mathcal{F},(A,y_A,\theta)}[X_{A_1}, X_{A_2}, X_F = 0] &= \beta^\oplus[Y_{A_1 \oplus A_2} = 1, X_{A_2}, X_{A_1}] \otimes \begin{bmatrix} \exp[\theta[L = 1]] \\ \exp[\theta[L = 1]] \end{bmatrix} [X_{A_1}] \\ &= \frac{1}{Z} \begin{bmatrix} 0 & \exp[\theta[L = 1]] \\ \exp[\theta[L = 1]] & 0 \end{bmatrix}\end{aligned}$$

for the normalization constant  $Z = 1 + 3 \cdot \exp[\theta[L = 1]]$  and

$$\mathbb{P}^{\mathcal{F},(A,y_A,\theta)}[X_{A_1}, X_{A_2}, X_F = 1] = \frac{1}{Z} \begin{bmatrix} 0 & 1 \\ \exp[\theta[L = 1]] & 0 \end{bmatrix}.$$

Also entailment can be checked for Hybrid Logic Network. Assuming a positive probability of all models of the integrated Hard Logic Network, the entailment can be checked only considering the Hard Logic Network. Here a query is a formula to retrieve information from a given network.

**Theorem 30 ( (Goessmann, 2025, Theorem 8.12))** *Let  $\mathbb{P}^{\mathcal{F},(A,y_A,\theta)}[X_{[d]}]$  be a Hybrid Logic Network. Given a query formula  $g$ , we have that  $\mathbb{P}^{\mathcal{F},(A,y_A,\theta)}[X_{[d]}] \models g$  if and only if*

$$f^{\mathcal{F},(A,y_A)} \models g,$$

where the tuple  $(A, y_A)$  denote the hard logic part of the network and

$$f^{\mathcal{F},(A,y_A)}[X_{[d]}] = \left( \bigwedge_{l \in A : y_l = 1} f_l[X_{[d]}] \right) \wedge \left( \bigwedge_{l \in A : y_l = 0} \neg f_l[X_{[d]}] \right).$$

**Example 16 Entailment for Accounting Logic** To check entailment of the query formula defined by

$$g[X_{A_1} = 0, X_{A_2} = 1, X_F = 1] = 1$$

and is set to zero otherwise, entailment can be checked by contracting

$$f^{\mathcal{F},(A,y_A)}[X_{A_1}, X_{A_2}, X_F] = X_{A_1} \oplus X_{A_2} \otimes \mathbb{I}$$

with  $g$  arriving at

$$\begin{aligned}&\langle f^{\mathcal{F},(A,y_A)}[X_{A_1}, X_{A_2}, X_F], \neg g[X_{A_1}, X_{A_2}, X_F] \rangle \\ &= \sum_{x_{A_1}, x_{A_2}, x_F \in \{0,1\}} f^{\mathcal{F},(A,y_A)}[X_{A_1} = x_{A_1}, X_{A_2} = x_{A_2}, X_F = x_F] (1 - g[X_{A_1} = x_{A_1}, X_{A_2} = x_{A_2}, X_F = x_F]) \\ &= \sum_{\substack{x_{A_1}, x_{A_2}, x_F \in \{0,1\} \\ (x_{A_1}, x_{A_2}, x_F) \neq (0,1,1)}} f^{\mathcal{F},(A,y_A)}[X_{A_1} = x_{A_1}, X_{A_2} = x_{A_2}, X_F = x_F] \\ &\geq f^{\mathcal{F},(A,y_A)}[X_{A_1} = 1, X_{A_2} = 0, X_F = 0] > 0.\end{aligned}$$

Therefore,  $\mathbb{P}^{\mathcal{F},(A,y_A,\theta)}[X_{[d]}]$  does not entail  $g$ . In this case, this is due to the fact, that  $g$  only assumes one model, despite the formula having multiple models. Doing the same calculations for

$$\tilde{g}[X_{A_1} = 1, X_{A_2} = 1, X_F = 1] = 0$$

and equal to 1 everywhere else leads to the constraction being equal to zero. Therefore,  $\mathbb{P}^{\mathcal{F},(A,y_A,\theta)}[X_{[d]}]$  does entail  $\tilde{g}$ .

## 6 Implementation

The architecture can be conveniently implemented with the python package `tnreason`. Multiple examples including graph-coloring, sat problems, Sudoku, and temporal clue are available<sup>1</sup>. To emphasize the intuitive implementation of CANs, a code snippet for the implementation of a CAN for the sat problem  $f[X_a = a, X_b = b, X_c = c] = (a \vee b) \wedge \neg c$  described in example 11 is explained. This propositional formula is encoded by a dictionary of variables encoded by nested lists, that need to all be fulfilled.

```
expressionsDict = {"f0" :  ["and", ["or", "a", "b"], ["not", "c"]]}
```

After importing the package by

```
from tnreason import representation, application, engine
```

the CAN can then be build by defining all cores. Activation cores then have suffices `_aC` and computation cores are denoted with suffices `_cC` by

```
cores = application.create_cores_to_expressionsDict(expressionsDict)
computationCores = {key: value for key, value in cores.items()
                    if key.endswith("_cC")}
```

The generated `computationCores` is a dictionary with the following keys and tensors of given shapes as expected in example 11.

```
'f0_aC', [2]
'(and_(or_a_b)_(not_c))_cC', [2, 2, 2]
'(or_a_b)_cC', [2, 2, 2]
'(not_c)_cC', [2, 2]
```

Based on this dictionary, the Computation-Activation Network can be build by setting the activation network for the single output feature (the output of  $f$ ) to a vector acting on the output of the basis encoding of  $f$ . Here a `SingleHybridFeature` is used, which allows for hard or sift activations. Then the value for the desired output of the feature is set to `True`.

```
caNet = representation.ComputationActivationNetwork(
    computationCoreDict=computationCores,
    featureDict={"f0" : representation.SingleHybridFeature(
        featureColor="(and_(or_a_b)_(not_c))_cV"}},
    canParamDict={"f0" : True}
)
```

---

1. <https://github.com/EnexaProject/enexa-tensor-reasoning/tree/version1>

As in example 11, the CAN then has the following form.

The other representation in example 11 can also be implemented. Both architectures can be found the linked notebook<sup>2</sup>. Note that more efficient representations of this network are possible and the one described here is mainly for pedagogic purposes. Furthermore the right implementation of the formula can be checked by contractions.

```
allCores = caNet.create_cores()
formula = engine.contract(coreDict = allCores,
                          openColors=["a_dV", "b_dV", "c_dV"])
assert formula[{"a_dV": 1, "b_dV" : 1, "c_dV" : 0}] == 1
assert formula[{"a_dV": 1, "b_dV" : 1, "c_dV" : 1}] == 0
```

The notebook also shows how the network can be normalized to represent a uniform probability distribution over all models of the formula.

```
distribution = engine.normalize(coreDict = allCores,
                               outColors = ["a_dV", "b_dV", "c_dV"],
                               inColors = [])
assert distribution[{"a_dV": 1, "b_dV" : 1, "c_dV" : 0}] == 1/3
assert distribution[{"a_dV": 1, "b_dV" : 1, "c_dV" : 1}] == 0
```

## 7 Conclusion& Outlook

- Conclusion: tensor representation close mathematical structures -¿ strength to do analysis? (Janina)
- Contraction algorithms (Alex)
- Max: Combine with foundation models / LLMs for agentic models (Max)

This work has treated the representation of several models in tensor networks. Model inference such as the computation of marginal distributions and the decision of entailment are formulated by tensor network contractions. These contraction can become bottlenecks, which are known as

- tree-widths of graphical models Pearl
- intractability of generic logical reasoning Russell and Norvig

Approximation schemes can be derived based on variational inference Wainwright and Jordan, such as loopy message passing schemes and mean field methods. Further frequently applied schemes are particle-based inference schemes such as Gibbs sampling.

Message-passing schemes appear in particular as belief propagation in probability theory and syntactical inference algorithms in logics. We can understand them as approximation of (potentially intractible) contractions and will dedicate future work to study them in the tntreasonformalism.

---

2. <https://colab.research.google.com/drive/14knFuMJHI683DAmUgJ-G10MQFueoXR6q#scrollTo=vr0YBVizhX2S>

## **Acknowledgments and Disclosure of Funding**

We acknowledge funding through BMBF (QROM) and MATH+ (Project PaA-7).

## References

- Ian Affleck, Tom Kennedy, Elliott H. Lieb, and Hal Tasaki. Rigorous results on valence-bond ground states in antiferromagnets. *Phys. Rev. Lett.*, 59:799–802, Aug 1987. doi: 10.1103/PhysRevLett.59.799. URL <https://link.aps.org/doi/10.1103/PhysRevLett.59.799>.
- Alejandro Mata Ali. Explicit solution equation for every combinatorial problem via tensor networks: MeLoCoToN. URL <http://arxiv.org/abs/2502.05981>.
- Artur S. Avila Garcez and Gerson Zaverucha. The connectionist inductive learning and logic programming system. 11(1):59–77. ISSN 1573-7497. doi: 10.1023/A:1008328630915. URL <https://doi.org/10.1023/A:1008328630915>.
- Samy Badreddine, Artur d’Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. 303:103649. ISSN 0004-3702. doi: 10.1016/j.artint.2021.103649. URL <https://www.sciencedirect.com/science/article/pii/S0004370221002009>.
- Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020. ISSN 1566-2535. doi: 10.1016/j.inffus.2019.12.012. URL <https://www.sciencedirect.com/science/article/pii/S1566253519308103>.
- A. V. Berezutskii, I. A. Luchnikov, and A. K. Fedorov. Simulating quantum circuits using the multi-scale entanglement renormalization ansatz. *Phys. Rev. Res.*, 7:013063, Jan 2025. doi: 10.1103/PhysRevResearch.7.013063. URL <https://link.aps.org/doi/10.1103/PhysRevResearch.7.013063>.
- P. Clifford and J. M. Hammersley. Markov fields on finite graphs and lattices. URL <https://ora.ox.ac.uk/objects/uuid:4ea849da-1511-4578-bb88-6a8d02f457a6>.
- Brandon C. Colelough and William Regli. Neuro-symbolic ai in 2024: A systematic review. 2025. URL <https://arxiv.org/abs/2501.05435>.
- Pedro Domingos. Tensor logic: The language of AI. URL <http://arxiv.org/abs/2510.12269>.
- Martin Eigel, Max Pfeffer, and Reinhold Schneider. Adaptive stochastic galerkin fem with hierarchical tensor representations. *Numerische Mathematik*, 136(3):765–803, 2017. doi: 10.1007/s00211-016-0850-x. URL <https://doi.org/10.1007/s00211-016-0850-x>.
- Artur d’Avila Garcez, Marco Gori, Luis C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. URL <http://arxiv.org/abs/1905.06088>.
- Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning*. MIT Press. ISBN 978-0-262-53868-8.

- Ivan Glasser, Ryan Sweke, Nicola Pancotti, Jens Eisert, and Ignacio Cirac. Expressive power of tensor-network factorizations for probabilistic modeling. 32.
- Alex Goessmann. *The tensor network approach towards efficient and explainable AI*. 2025.
- Alex Goeßmann, Ingo Roth, Gitta Kutyniok, Michael Götte, Ryan Sweke, and Jens Eisert. Tensor network approaches for data-driven identification of non-linear dynamical laws. In *Advances in Neural Information Processing Systems - First Workshop on Quantum Tensor Networks in Machine Learning*, page 21.
- Alex Christoph Goeßmann. Uniform concentration of tensor and neural networks: An approach towards recovery guarantees. URL <https://depositonce.tu-berlin.de/handle/11303/15990>.
- Nikita Gourianov, Peyman Givi, Dieter Jaksch, and Stephen B. Pope. Tensor networks enable the calculation of turbulence probability distributions. *Science Advances*, 11(5), January 2025. ISSN 2375-2548. doi: 10.1126/sciadv.ads5990. URL <http://dx.doi.org/10.1126/sciadv.ads5990>.
- Wolfgang Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*. Springer Series in Computational Mathematics. Springer-Verlag. ISBN 978-3-642-28026-9. doi: 10.1007/978-3-642-28027-6.
- Wolfgang Hackbusch and Stefan Kühn. A new scheme for the tensor representation. *The journal of Fourier analysis and applications*, 15(5):706–722, 2009. ISSN 1069-5869. doi: 10.1007/s00041-009-9094-9.
- Paul Hagemann, Janina Schütte, David Sommer, Martin Eigel, and Gabriele Steidl. Sampling from boltzmann densities with physics informed low-rank formats. In Tatiana A. Bubba, Romina Gaburro, Silvia Gazzola, Kostas Papafitsoros, Marcelo Pereyra, and Carola-Bibiane Schönlieb, editors, *Scale Space and Variational Methods in Computer Vision*, page 374–386, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-92366-1.
- Frank Lauren Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6:164–189, 1927. URL <https://api.semanticscholar.org/CorpusID:124183279>.
- Sepp Hochreiter. Toward a broad AI. 65(4):56–57. ISSN 0001-0782, 1557-7317. doi: 10.1145/3512715. URL <https://dl.acm.org/doi/10.1145/3512715>.
- Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 1. edition edition. ISBN 978-0-262-01319-2.
- J. Landsberg. *Tensors: Geometry and Applications*, volume 128 of *Graduate Studies in Mathematics*. American Mathematical Society. ISBN 978-0-8218-6907-9 978-0-8218-8481-2 978-0-8218-8483-6 978-1-4704-0923-4.
- Zachary C. Lipton. The mythos of model interpretability, 2017. URL <https://arxiv.org/abs/1606.03490>.

- Giuseppe Marra, Sebastijan Dumančić, Robin Manhaeve, and Luc De Raedt. From statistical relational to neurosymbolic artificial intelligence: A survey. 328:104062. ISSN 0004-3702. doi: 10.1016/j.artint.2023.104062. URL <https://www.sciencedirect.com/science/article/pii/S0004370223002084>.
- John McCarthy. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 75–91. Her Majesty’s Stationary Office. URL <http://www-formal.stanford.edu/jmc/mcc59.html>.
- Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. 104(1):11–33. ISSN 0018-9219, 1558-2256. doi: 10.1109/JPROC.2015.2483592. URL <https://ieeexplore.ieee.org/document/7358050/>.
- Román Orús. Tensor networks for complex quantum systems. *Nature Reviews Physics*, 1(9):538–550, August 2019. ISSN 2522-5820. doi: 10.1038/s42254-019-0086-7. URL <http://dx.doi.org/10.1038/s42254-019-0086-7>.
- I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5): 2295–2317, 2011. doi: 10.1137/090752286. URL <https://doi.org/10.1137/090752286>.
- Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann. ISBN 978-1-55860-479-7.
- Roger Penrose. *Spinors and Space-Time: Volume 1, Two-Spinor Calculus and Relativistic Fields*. Cambridge University Press. ISBN 978-0-521-33707-6.
- Matthew Richardson and Pedro Domingos. Markov logic networks. 62(1):107–136. ISSN 0885-6125, 1573-0565. doi: 10.1007/s10994-006-5833-1. URL <http://link.springer.com/10.1007/s10994-006-5833-1>.
- Elina Robeva and Anna Seigal. Duality of graphical models and tensor networks. 8(2): 273–288. ISSN 2049-8772. doi: 10.1093/imaiai/iy009.
- Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach, Global Edition: A Modern Approach, Global Edition*. Pearson, 4 edition. ISBN 978-1-292-40113-3.
- Aaron Sander, Maximilian Fröhlich, Martin Eigel, Jens Eisert, Patrick Gelß, Michael Hintermüller, Richard M. Milbradt, Robert Wille, and Christian B. Mendl. Large-scale stochastic simulation of open quantum systems. URL <http://arxiv.org/abs/2501.17913>.
- Aaron Sander, Maximilian Fröhlich, Mazen Ali, Martin Eigel, Jens Eisert, Michael Hintermüller, Christian B. Mendl, Richard M. Milbradt, and Robert Wille. Quantum circuit simulation with a local time-dependent variational principle, 2025. URL <https://arxiv.org/abs/2508.10096>.
- Md Kamruzzaman Sarker, Lu Zhou, Aaron Eberhart, and Pascal Hitzler. Neuro-symbolic artificial intelligence: Current trends. 34(3):197–209. ISSN 18758452, 09217126. doi:

10.3233/AIC-210084. URL <https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/AIC-210084>.

Geoffrey G. Towell and Jude W. Shavlik. Knowledge-based artificial neural networks. 70 (1):119–165. ISSN 0004-3702. doi: 10.1016/0004-3702(94)90105-8. URL <https://www.sciencedirect.com/science/article/pii/0004370294901058>.

Martin J. Wainwright and Michael Irwin Jordan. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc. ISBN 978-1-60198-184-4.

Steven R. White. Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.*, 69:2863–2866, Nov 1992. doi: 10.1103/PhysRevLett.69.2863. URL <https://link.aps.org/doi/10.1103/PhysRevLett.69.2863>.