Journal of Machine Learning Research 23 (2022) 1-44 Submitted 1/21; Revised 5/22; Published 9/22

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

Alex Goessmann ALEX.GOESSMANN@WIAS-BERLIN.DE
Janina Schütte JANINA.SCHUETTE@WIAS-BERLIN.DE
Maximilian Fröhlich MAXIMILIAN.FROEHLICH@WIAS-BERLIN.DE
Martin Eigel MARTIN.EIGEL@WIAS-BERLIN.DE
Weierstrass Institute of Applied Analysis and Stochastic
Anton-Wilhelm-Amo-Straße 39
Berlin, 10117, Germany

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.

©2022 Author One and Author Two.

License: CC-BY 4.0, see `https://creativecommons.org/licenses/by/4.0/`. Attribution requirements are provided at `http://jmlr.org/papers/v23/21-0000.html`.

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 grpahical 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 combinatorical 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 $\tau[X_0, \ldots, X_{d-1}]$ of order d and with leg dimensions $m_0, \ldots, m_{d-1}$ is defined through its coordinates*

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

for index tuples

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

Tensors $\tau[X_0, \ldots, X_{d-1}]$, also denoted by $\tau\left[X_{[d]}\right]$, 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 $\tau\left[X_{[d]}\right]$ boolean, when $\mathrm{im}(\tau) \subset [2]$, 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*

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

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

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

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*

$$\epsilon_x[X = \tilde{x}] = \begin{cases} 1 & \textit{if} \quad x = \tilde{x} \\ 0 & \textit{else}\,. \end{cases} \tag{1}$$

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

$$\epsilon_x[X] = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}^T, \tag{2}$$

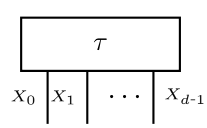
where the 1 is at the x-th coordinate of the vector.

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

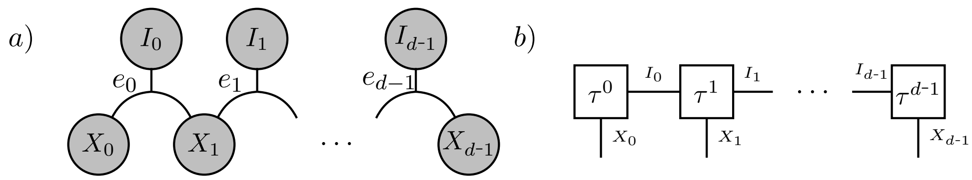
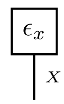


Figure 1: Hypergraph to a TT format. a) Node-centric design. b) Corresponding tensor-network on the edges of the hypergraph.



Assigning x to the categorial variable X will retrieve the xth 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*

  $$\{e_0 = (X_0, I_0)\} \cup \{e_k = (I_{k-1}, X_k, I_{k+1}) \,:\, k \in \{1, \dots, d-2\}\} \cup \{e_{m-1} = (I_{d-2}, X_{d-1})\}$$

  each decorated by a tensor of order 3 (respectively 2 for $k \in \{0, p-1\}$).

Example 3 (The CP **format)** *The Candecomp-Parafac (*CP *(Hitchcock) 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 $\tau^{e_k} [X_k, I]$.

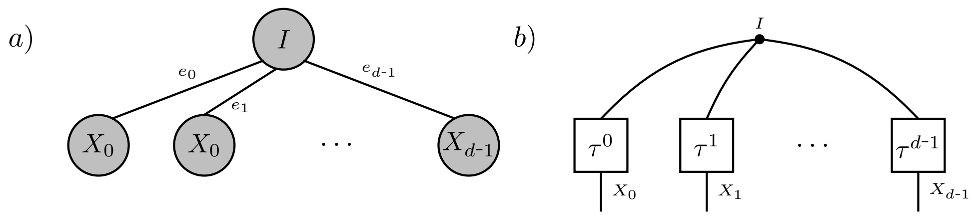


Figure 2: Hypergraph to a CP format. a) Node-centric design. b) Corresponding tensor-network on the edges of the hypergraph.

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*

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

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

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

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

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

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.

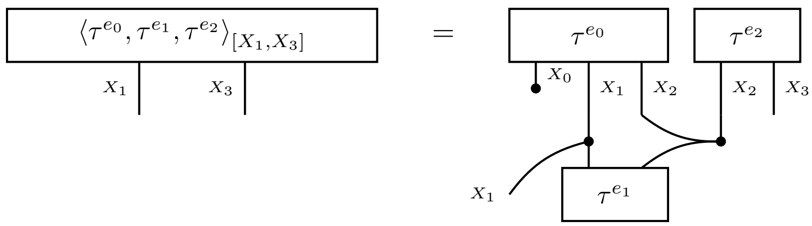


Figure 3: Graphical depiction of a tensor network contraction with the open variables X_1, X_3.

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

$$\left\langle \tau^{\mathcal{G}} \right\rangle_{[X_{\mathcal{U}}]} \in \bigotimes_{v \in \mathcal{U}} \mathbb{R}^{m_v} \tag{3}$$

defined coordinatewise by the sum

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

We call $X_{\mathcal{U}}$ 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*

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

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

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

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

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

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

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

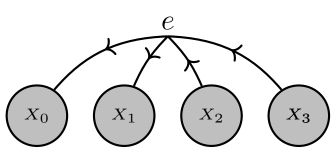
The normalization of an on $\mathcal{V}^{\text{in}} \subset \mathcal{V}$ normalizable tensor is the tensor

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

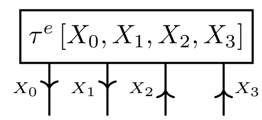
where $\mathcal{V}^{\text{out}} = \mathcal{V}/\mathcal{V}^{\text{in}}$.

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

a)



b)



2.5 Function encoding and Computation-Activation Networks

Might need to discuss image enumerating maps!

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*

$$q : \bigtimes_{k \in [d]} [m_k] \rightarrow \bigtimes_{l \in [r]} [m_l]$$

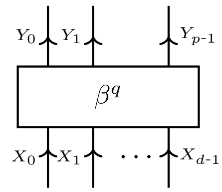
between these state sets. Then the basis encoding of q is a tensor

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

defined by

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

Basis encodings are directed tensors (see (Goessmann, 2025, Theorem 14.10)) and are thus depicted as decorations of directed edges in hypergraphs:



The entries of the basis encoding are defined by

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

Based on these concepts the main architecture can be defined.

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

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

We refer to any member $\mathbb{P}\left[X_{[d]}\right]\in\Lambda^{t,\mathcal{G}}$ 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 ξ to represent the tensor to contract with (also called *activation network*).

3 Decomposition of Probability Distributions

We here investigate tensor network decomposition mechanisms of probability distributions. 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. Further we present the Computation mechanism, in which the Fisher-Neyman Factorization Theorem is used to decompose distributions in the presence of sufficient statistics.

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*

$$\mathbb{P}\left[X_{[d]}\right]\,:\,\bigtimes_{k\in[d]}[m_k]\to\mathbb{R}$$

which is non-negative, that is for any $x_{[d]} \in \bigtimes_{k\in[d]}[m_k]$ *it holds*

$$\mathbb{P}\left[X_{[d]} = x_{[d]}\right] \geq 0\,,$$

and which is normalized, that is

$$\left\langle \mathbb{P}\left[X_{[d]}\right]\right\rangle_{[\varnothing]} = 1\,.$$

Let Z *be another variable taking values in a possibly infinite set* $\mathrm{val}(Z)$*. Then a tensor* $\mathbb{P}\left[X_{[d]}\middle|Z\right]$ *is a family of joint probability distributions, if for any* $z \in \mathrm{val}(Z)$ *the slice* $\mathbb{P}\left[X_{[d]}\middle|Z = z\right]$ *is a joint probability distribution.*

Example 4 (Family of independent Coin Tosses) *Consider tossing a coin with head probability* $z \in [0,1]$ *and repeating the experiment independently* $d \in \mathbb{N}$ *times. We define a variable* Z *taking values in* $\mathrm{val}(Z) = [0,1]$ *and denote by* $X_{[d]}$ d *boolean variables. Then the family of coin toss distributions is modeled by the tensor* $\mathbb{P}\left[X_{[d]}, Z\right]$ *with coordinates* $x_{[d]} \in \bigtimes_{k\in[d]}[2]$ *and* $z \in [0,1]$ *by*

$$\mathbb{P}\left[X_{[d]} = x_{[d]}, Z = z\right] = \prod_{k\in[d]} z^{x_k}(1-z)^{1-x_k} = z^{\sum_{i=1}^d x_i}(1-z)^{d-\sum_{i=1}^d x_i}\,.$$

Notice, that for each slice with respect to $z \in [0,1]$ *we have by the binomial theorem* $\left\langle \mathbb{P}\left[X_{[d]}, Z = z\right]\right\rangle_{[\varnothing]} = 1$ *and thus* $\mathbb{P}\left[X_{[d]}, Z\right]$ *is indeed a family of probability distributions. For* $d = 2$ *we have more explicitly for any* $z \in [0,1]$*:*

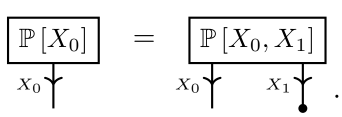
$$\mathbb{P}\left[X_{[2]}\middle|Z = z\right] = \begin{array}{cc} & \begin{array}{cc} & X_1 \\ 0 & & 1 \end{array} \\ X_0 \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} (1-z)^2 & z\cdot(1-z) \\ z\cdot(1-z) & z^2 \end{bmatrix} \end{array}\,.$$

A basic inference operation on probability distributions is the computation of marginal and conditional distribution.

Definition 11 *For any distribution* $\mathbb{P}\left[X, Y\right]$ *the marginal distribution is given by the contraction*

$$\mathbb{P}\left[X\right] := \left\langle \mathbb{P}\left[X, Y\right]\right\rangle_{[X]}$$

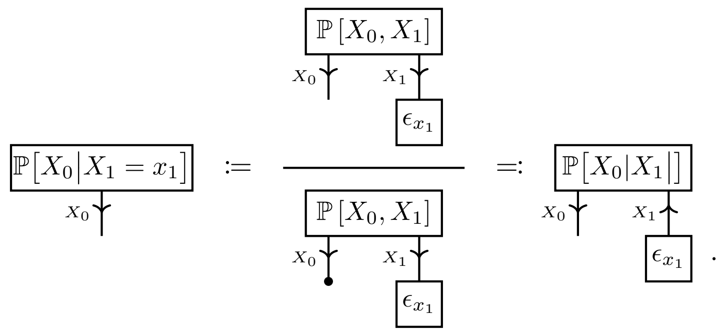
which is depicted by the diagram



The conditional distribution of X *on* Y *is a tensor* $\mathbb{P}\left[X\middle|Y\right]$ *defined for* $y \in [m_1]$

$$\mathbb{P}\left[X\middle|Y = y\right] := \begin{cases} \frac{1}{m}\cdot \mathbb{I}\left[X\right] & \textit{if} \quad \left\langle \mathbb{P}\left[X, Y = y\right]\right\rangle_{[\varnothing]} = 0 \\ \frac{1}{\left\langle \mathbb{P}\left[X, Y=y\right]\right\rangle_{[\varnothing]}} \cdot \mathbb{P}\left[X, Y = y\right] & \textit{else} \end{cases}$$

and in the second case depicted by the diagram



3.2 The Independence mechanism: Graphical Model Factorization

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

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

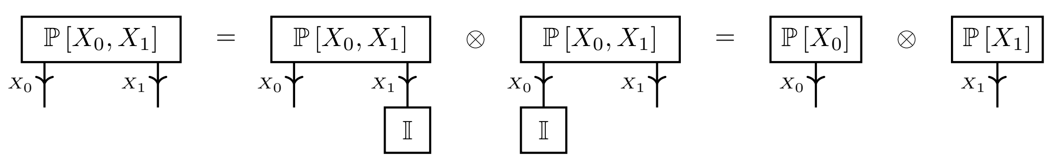
and therefore scales exponentially in the number of coordinates. To find efficient representation schemes of probability distributions by tensor networks, we need to exploit additional properties of the distribution. Independence leads to severe sparsifications of conditional probabilities and is therefore the key assumption to gain sparse decompositions of probability distributions.

Definition 12 (Independence) *We say that X_0 is independent of X_1 with respect to a distribution $\mathbb{P}[X_0, X_1]$, if the distribution is the tensor product of the marginal distributions, that is*

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

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

Thus, independence appears directly as a tensor–product decomposition of probability distribution. Using tensor network diagrams we depict this property by



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 $x_2 \in [m_2]$*

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

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

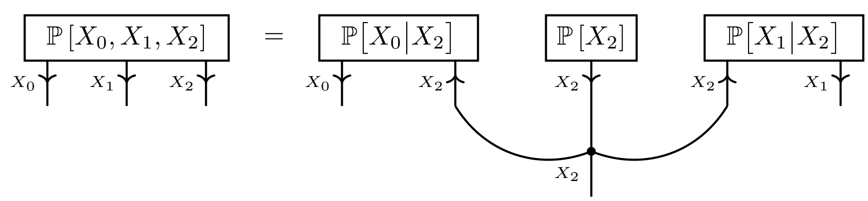
Conditional independence stated in Def. 13 has a close connection with independence stated in Def. 12. To be more precise, X_0 is independent of X_1 conditioned on X_2, if and only if X_0 is independent of X_1 with respect to any slice $\mathbb{P}\big[X_0, X_1 \big| X_2 = x_2\big]$ of the conditional distribution $\mathbb{P}\big[X_0, X_1 \big| X_2\big]$.

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

Corollary 14 *Let $\mathbb{P}[X_0, X_1, X_2]$ be a joint distribution. If and only if X_0 is independent of X_1 conditioned on X_2 the distribution satisfies*

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

In a diagrammatic notation this is depicted by



This conditional–independence pattern is the basic local building block that is generalized in Markov networks, which we define in the following.

Definition 15 (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_v defined by the tensor*

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

We call the denominator

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

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

We directly defined here graphical models on hypergraphs. It is known that probabilistic graphical models are dual to tensor networks Robeva and Seigal; Glasser et al.. We define graphical models 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).

We can interpret the factors $\tau\left[X_{[d]}\right]$ as activation cores placed on the hyperedges 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.

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 seperation properties, as we define next.

Definition 16 (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 17 (Clique-Capturing Hypergraph) *We call a hypergraph $\mathcal{G}$ clique-capturing, when each subset $\mathcal{U} \subset \mathcal{V}$ is contained in a hyperedge, if for any $a, b \in \mathcal{U}$ there is a hyperedge $e \in \mathcal{E}$ with $a, b \in \mathcal{U}$.*

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

Theorem 18 (Hammersley-Clifford Factorization Theorem) *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.*

Proof Shown in Appendix Sect. A. ■

Example 5 (I.i.d. Boolean Variables: Coin toss interpretation) *Let there be d boolean variables $X_{[d]}$, which are i.i.d. drawn from a positive distribution $\mathbb{P}[X]$. From the pairwise independencies of X_k it follows with the Hammersley-Clifford Factorization Theorem that the distribution is representable by an elementary tensor network, that is*

$$\mathbb{P}\left[X_{[d]}\right] = \bigotimes_{k\in[d]} \mathbb{P}\left[X_k\right] .$$

Equivalently, Thm. 18 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

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

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

Definition 19 *Let $\mathbb{P}[X, Z]$ be a joint distribution of the m-dimensional variable X and the n-dimensional variable Z and let*

$$t : [m] \to [n]$$

be a statistic. We are interested in the distribution $\mathbb{P}[X, Z, Y_t] = \left\langle \mathbb{P}[X, Z], \beta^t[Y_t, X]\right\rangle_{[X,Z,Y_t]}$. We say that t is a sufficient statistic for Z if and only if X is independent of Z conditioned on Y_t.

Note that the independence in Def. 19 is true if and only if

$$\mathbb{P}\left[X\middle|Z, Y_t\right] = \mathbb{P}\left[X\middle|Y_t\right] \otimes \mathbb{I}[Z]\ .$$

Example 6 (Sufficient Statistics for the Probability) *Let Z be the value $\mathbb{P}\left[X_{[d]} = x_{[d]}\right]$, when drawing $X_{[d]}$ from $\mathbb{P}\left[X_{[d]}\right]$. Then t is a sufficient statistic for $Z = \mathbb{P}\left[X_{[d]}\right]$, if for all y in the image of t we have*

$$\mathbb{P}\left[X_{[d]} = x_{[d]}\middle|t\left(x_{[d]}\right) = y\right] = \begin{cases} \frac{1}{\left|\{x_{[d]}\,:\,t\left(x_{[d]}\right)=y\}\right|} & \textit{if} \quad t(x_{[d]}) = y \\ 0 & \textit{else} \end{cases}.$$

When knowing the value $tx_{[d]}$ of the sufficient statistic at a given index $x_{[d]}$, we then also know the probability $\mathbb{P}\left[X_{[d]} = x_{[d]}\right]$. The function t is thus a sufficient statistic for $Z = \mathbb{P}\left[X_{[d]}\right]$, if and only if there is a tensor $\xi\left[Y_{[p]}\right]$ with

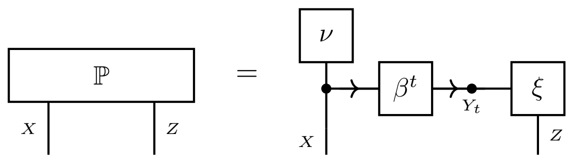
$$\mathbb{P}\left[X_{[d]}\right] = \left\langle \beta^t\left[Y_{[p]}, X_{[d]}\right], \xi\left[Y_{[p]}\right]\right\rangle_{\left[X_{[d]}\right]}\ .$$

Example 6 hints at a connection between sufficient statistics and decompositions into CompAct Nets. More generally, such decompositions are provided by the Fisher-Neyman Factorization Theorem.

Theorem 20 (Fisher-Neyman Factorization Theorem) *Let $\mathbb{P}$ be a joint distribution of variables X, Z with values $\mathrm{val}(X)$, $\mathrm{val}(Z)$. Let there further be a finite set $\mathrm{val}(Y_t)$. Then $t : \mathrm{val}(X) \rightarrow \mathrm{val}(Y_t)$ is a sufficient statistic for Z if and only if there are tensors $\nu[X]$ and $\xi[Y_t, Z]$ such that*

$$\mathbb{P}[X, Z] = \left\langle \xi[Y_t, Z] \, \beta^t[Y_t, X], \nu[X] \right\rangle_{[X,Z]} .$$

We depict this equation diagrammatically by



.

Proof Shown in more generality in the appendix, see Thm. 36. ■

Notice, that the definition of sufficient statistic does not make use of the marginal distribution $\mathbb{P}[Z]$. We therefore can define sufficient statistics also for families of distributions $\mathbb{P}[X|Z]$, with respect to arbitrary non-degenerate marginal distribution $\mathbb{P}[Z]$. We then use the Thm. 20 to embed such families in CompAct Nets.

Corollary 21 *Let $\mathbb{P}[X_{[d]}|Z]$ be an arbitrary family of distributions of $X_{[d]}$, which is . Then there is a tensor $\nu\left[X_{[d]}\right]$ and a activation tensors $\xi\left[Y_{[p]}, Z\right]$ such that for any $z \in \mathrm{val}(Z)$*

$$\mathbb{P}\left[X_{[d]} \middle| Z = z\right] \in \Lambda^{t,\mathrm{MAX},\nu} .$$

Example 7 (Order Statistic for Boolean Variables: Coin toss interpretation) *Let there be d boolean variables $X_{[d]}$ and a family $\{\mathbb{P}^\theta\left[X_{[d]}\right] : \theta \in \Theta\}$ of distributions. The order statistic assigns to each tuple $x_{[d]}$ the ordered tuple, which effectively counts the number of 1 coordinates in the tuple $x_{[d]}$, that is the statistic*

$$t^+ : \bigtimes_{k\in[d]} [m_k] \rightarrow [p] \quad , \quad t^+(x_{[d]}) = |\{k : x_k = 1\}| .$$

When the order statistic is sufficient, the detailed order of the outcomes in uninformative about the member $\theta \in \Theta$ from which the random variables have been drawn. Let us now investigate those families for which t^+ is a sufficient statistic. By the Fisher-Neyman Factorization Theorem Thm. 20 t^+ is a sufficient statistic if and only if there are tensors $\nu\left[X_{[d]}\right]$ and $\xi[Y_+, \Theta]$ such that for each $\theta \in \Theta$

$$\mathbb{P}^\theta\left[X_{[d]}\right] = \left\langle \xi^\theta[Y_+], \beta^{t^+}\left[Y_+, X_{[d]}\right], \nu\left[X_{[d]}\right] \right\rangle_{\left[X_{[d]}\right]} .$$

The family of distributions, such that the variables $X_{[d]}$ are i.i.d. with respect to each (see Example 5) are the special case, where $\nu\left[X_{[d]}\right] = \mathbb{I}\left[X_{[d]}\right]$ and the family is labeled by $\theta \in [0, 1]$ such that for $\theta \in (0, 1)$ and $k \in [d + 1]$

$$\xi^\theta[Y_+ = k] = \theta^{d-k} \cdot \theta^k ,$$

and for $\theta \in \{0,1\}$

$$\xi^{\theta}\left[Y_{+}\right] = \begin{cases} \epsilon_{0}\left[Y_{+}\right] & \textit{if} \quad \theta = 0 \\ \epsilon_{d}\left[Y_{+}\right] & \textit{if} \quad \theta = 1 \end{cases} .$$

The marginal distribution $\mathbb{P}^{\theta}\left[Y_{+}\right]$ *is then the binominal distribution* $B(d,\theta)$.

The Fisher-Neyman Theorem is the fundamental motivation for the CompAct Nets Architecture:

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

Example 8 (Graphical Models as a special case of CompAct Nets) *Recall Def. 9. Given a statistic* $t : \bigtimes_{k\in[d]}[m_k] \rightarrow \mathbb{R}^p$ *and a hypergraph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *on the image coordinates* $Y_{[p]}$, *any by* t *computable and by* $\mathcal{G}$ *activated CompAct Nets has the form*

$$\mathbb{P}\left[X_{[d]}\right] = \left\langle \xi\left[Y_{[p]}\right], \beta^{t}\left[Y_{[p]}, X_{[d]}\right] \right\rangle_{\left[X_{[d]}|\varnothing\right]}$$

where $\xi\left[Y_{[p]}\right]$ *is an arbitrary non-negative tensor. For graphical models we take the* identity statistic

$$\delta\left(x_{[d]}\right) = x_{[d]},$$

so that the image coordinates coincide with the variables and there are no non-trivial computation cores. The associated basis encoding is just the identity tensor

$$\beta^{\delta}\left[Y_{[d]}, X_{[d]}\right] = \delta\left[X_{[d]}, Y_{[d]}\right] .$$

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

$$\mathbb{P}\left[X_{[d]}\right] = \left\langle \xi\left[Y_{[p]}\right], \beta^{\delta}\left[Y_{[d]}, X_{[d]}\right] \right\rangle_{\left[X_{[d]}|\varnothing\right]} \qquad = \left\langle \xi\left[X_{[d]}\right] \right\rangle_{\left[X_{[d]}|\varnothing\right]}$$

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.

3.4 Exponential families in case of elementary activation tensors

A classical theorem by Pitman-Koopman-Darmois (see **?**) states, that whenever a family with constant support and a finite sufficient statistic for arbitrary large data sets is in an exponential family. We now restrict the activation cores to specific elementary tensors, which correspond with further assumptions on the dependence of $\mathbb{P}$ and t made by exponential families. For a discussion of further universal properties of exponential families, such that the existence of priors and entropy maximizers, see Murphy.

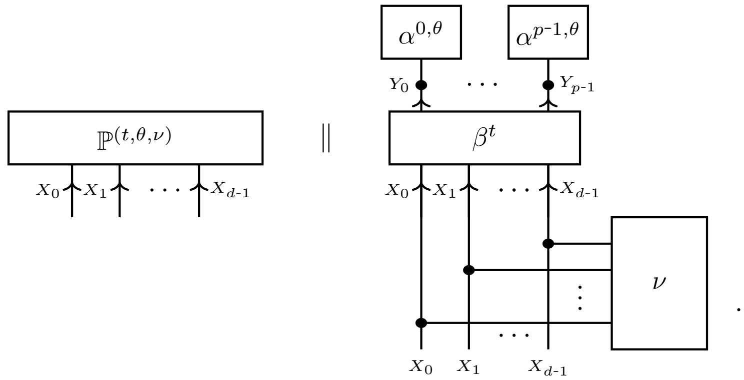


Figure 4: Tensor Network diagram of a member of an exponential family $\mathbb{P}^{t,\nu}\left[X_{[d]}|\Theta=\theta\right]$ before normalization as an CompAct Net with elementary activation, that is an element in $\Lambda^{t,\mathrm{EL},\nu}$.

Definition 22 (Exponential Family) *Given a base measure ν and a statistic $t : \bigtimes_{k\in[d]}[m_k] \to \mathbb{R}^p$ we enumerate for each coordinate $l \in [p]$ the image $\mathrm{im}\,(t_l)$ by an interpretation map*

$$I_l \,:\, [|\mathrm{im}\,(t_l)|] \to \mathrm{im}\,(t_l)\ .$$

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

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

and define the distribution

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

We then call the tensor $\mathbb{P}^{t,\nu}\left[X_{[d]}|\Theta\right]$ with $\mathrm{val}\,(\Theta)=\mathbb{R}^p$ and slices for $\theta\in\Gamma$ by

$$\mathbb{P}^{t,\nu}\left[X_{[d]}|\Theta=\theta\right] = \mathbb{P}^{(t,\theta,\nu)}\left[X_{[d]}\right]$$

the exponential family to the statistic t and the base measure ν.

Note that by construction each member of an exponential family is an element in a CompAct Net with elementary activation cores (see Def. **??**), that is

$$\mathbb{P}^{t,\nu}\left[X_{[d]}|\Theta=\theta\right] \in \Lambda^{t,\mathrm{EL},\nu}\,.$$

Example 9 (Joint distributions of two booleans) *In general, joint distribution of two Boolean variables X_0, X_1 are 2x2 matrices of non-negative coordinates summing to 1:*

$$\mathbb{P}\left[X_{[2]}\right] = \begin{bmatrix} p_{0,0} & p_{0,1} \\ p_{1,0} & p_{1,1} \end{bmatrix}$$

In the following example, we will assume at different points that X_0, X_1 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.

$$\mathbb{P}\left[X_{[2]}\right] = \begin{bmatrix} p_{0,0} & p_{0,1} \\ p_{1,0} & 1-(p_{0,0}+p_{0,1}+p_{1,0}) \end{bmatrix}$$

Let us now restrict to those distributions, which have the sum $X_0 + X_1$ 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.

$$\mathbb{P}\left[X_{[2]}\right] = \begin{bmatrix} p_{0,0} & p_{0,1} \\ p_{0,1} & 1-p_{0,0}-2p_{0,1} \end{bmatrix}$$

This symmetry also implies, that the distributions are identically distributed, i.e. for any $x \in \{0,1\}$ we have

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

Restricting further to those, where X_0 and X_1 are independent and the distribution is everywhere supported, brings us to the rank one formulation of the distribution

$$\mathbb{P}\left[X_0, X_1\right] = \begin{bmatrix} \mathbb{P}\left[X_0=0\right]\mathbb{P}\left[X_1=0\right] & \mathbb{P}\left[X_0=0\right]\mathbb{P}\left[X_1=1\right] \\ \mathbb{P}\left[X_0=1\right]\mathbb{P}\left[X_1=0\right] & \mathbb{P}\left[X_0=1\right]\mathbb{P}\left[X_1=1\right] \end{bmatrix} = \mathbb{P}\left[X_0\right] \otimes \mathbb{P}\left[X_1\right]$$

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

$$\mathbb{P}\left[X_0\right] = \frac{1}{1+\exp\left[\theta\right]} \begin{bmatrix} 1 \\ \exp\left[\theta\right] \end{bmatrix}$$

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

$$\mathbb{P}\left[X_0, X_1\right] = \frac{1}{(1+\exp\left[\theta\right])^2} \begin{bmatrix} 1 \\ \exp\left[\theta\right] \end{bmatrix} \begin{bmatrix} 1 & \exp\left[\theta\right] \end{bmatrix}$$

We can interpret this distribution as two independent coin tosses with outcome X_0 and X_1 and head probability

$$\mathbb{P}\left[X_0 = 1\right] = \mathbb{P}\left[X_1 = 1\right] = \frac{\exp\left[\theta\right]}{1+\exp\left[\theta\right]}$$

which is the sigmoid of θ and inverted by the logit

$$\theta = \ln\left[\frac{\mathbb{P}\left[X_0=1\right]}{1-\mathbb{P}\left[X_0=1\right]}\right] .$$

Consistent with the above parametrization, we have a uniform distribution of X_0 and X_1 in the fair coin toss case $\mathbb{P}\left[X_0 = 1\right] = 0.5$, where $\theta = 0$.

As a Computation-Activation Network we can represent any distribution $\mathbb{P}[X_0, X_1]$ *with the head count* $+$ *as sufficient statistic by*

$$\mathbb{P}[X_0, X_1] = \left\langle \beta^+ [Y_+, X_0, X_1], \xi [Y_+] \right\rangle_{[X_0, X_1|\varnothing]} ,$$

such that

$$\begin{aligned}\mathbb{P}[X_0 = x_0, X_1 = x_1] &= \frac{1}{Z} \left\langle \beta^+ [Y_+, X_0, X_1], \xi [Y_+] \right\rangle_{[X_0=x_0, X_1=x_1]} \\ &= \frac{1}{Z} \sum_{y_+ \in [2]} \beta^+ [Y_+ = y_+, X_0 = x_0, X_1 = x_1] \cdot \xi [Y_+ = y_+] \\ &= \frac{1}{Z} \xi [Y_+ = x_0 + x_1] ,\end{aligned}$$

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

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

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 $p_{0,1} = \mathbb{P}[X_0 = 0] \cdot \mathbb{P}[X_0 = 1] = (1+\exp[\theta])^{-1} \cdot \exp[\theta](1+\exp[\theta])^{-1}$ *and* $p_{1,1} = (\exp[\theta](1+\exp[\theta])^{-1})^2$.

4 Decompositions based on Propositional Syntax

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.

4.1 Propositional Semantics by Boolean Tensors

Definition 23 *A* propositional formula $f[X_{[d]}]$ *depending on* d *atoms* X_k *is a boolean-valued tensor*

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

We call a state $x_{[d]} \in \bigtimes_{k \in [d]} [2]$ *a* model *of a propositional formula* f*, if*

$$f[X_{[d]} = x_{[d]}] = 1 ,$$

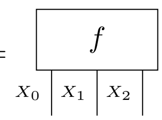
where we associate True $\leftrightarrow 1$ *and False* $\leftrightarrow 0$*. If there is a model to a propositional formula, we say the formula is* satisfiable*.*

Example 10 *Let there be $d = 3$ boolean variables $X_{[3]}$ and a propositional formula*

$$f\left[X_{[3]}\right] = (X_0 \vee X_1) \wedge \neg X_2 \,.$$

In a graphical depiction and in the coordinatewise representation this formula can be represented as

$$f\left[X_{[3]}\right] =$$



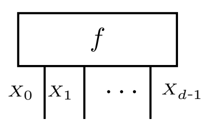
$$= \begin{array}{cc} & \overset{X_1}{\begin{matrix} 0 & 1 \end{matrix}} \\ X_0 \begin{matrix} 0 \\ 1 \end{matrix} & \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \end{array} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \underset{X_2}{0 \to 1} \,.$$

In the state set $\bigtimes_{k\in[d]}[2] = \{0,1\} \times \{0,1\} \times \{0,1\}$ we have three models of the formula by the positions of the non-zero entries in the tensor, i.e. $f\left[X_{[3]} = x_{[3]}\right] = 1$ if and only if

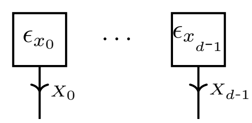
$$x_{[3]} \in \{(1,0,0),(0,1,0),(1,1,0)\} \,.$$

The formula f is therefore satisfiable.

CP decomposition Since the tensor $f\left[X_{[d]}\right]$ 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.



$$= \sum_{\substack{x_0,\dots,x_{d-1} \in \bigtimes_{k\in[d]}[2] \\ f(x_0,\dots,x_{d-1}) = 1}}$$



This decomposition corresponds to the CP decomposition of a tensor.

Example 11 *For the formula described in Example* **??**, *we have*

$$\begin{aligned} f\left[X_{[3]}\right] = &(\epsilon_1[X_0] \otimes \epsilon_0[X_1] \otimes \epsilon_0[X_2]) + (\epsilon_0[X_0] \otimes \epsilon_1[X_1] \otimes \epsilon_0[X_2]) \\ &+ (\epsilon_1[X_0] \otimes \epsilon_1[X_1] \otimes \epsilon_0[X_2]), \end{aligned}$$

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

$$\begin{aligned} f\left[X_{[3]} = x_{[3]}\right] &= (\epsilon_1[X_0 = 1] \otimes \epsilon_0[X_1 = 1] \otimes \epsilon_0[X_2 = 0]) \\ &\quad + (\epsilon_0[X_0 = 1] \otimes \epsilon_1[X_1 = 1] \otimes \epsilon_0[X_2 = 0]) \\ &\quad + (\epsilon_1[X_0 = 1] \otimes \epsilon_1[X_1 = 1] \otimes \epsilon_0[X_2 = 0]) \\ &= 1 \cdot 0 \cdot 1 + 0 \cdot 1 \cdot 1 + 1 \cdot 1 \cdot 1 = 1. \end{aligned}$$

Model counts by contraction Each coordiante of the propositional formula is either a 1 or 0 encoding if the indexed state is a model of the formula or not. In this way, the contraction $\langle f \rangle_{[\varnothing]}$ counts the number of models of the propositional formula f. One can therefore decide the satisfiability of a formula by checking if $\langle f \rangle_{[\varnothing]} > 0$.

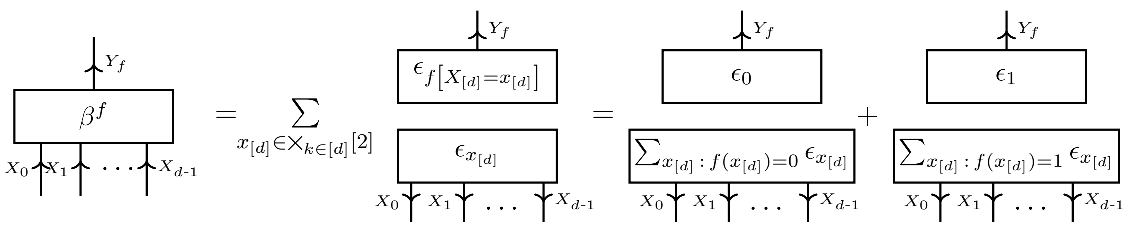
Basis encoding Representing booleans by elements in $\{0,1\}$ leads to the problem, that negation is an affine transformation and can not be represented by multilinear tensors (Goessmann, 2025, Section 4.1.1). Therefore, instead of using this *coordinate calculus* an approach based on *basis calculus* is employed, which is explained in this section. 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 Def. 2. Propositional formulas f can be expressed in terms of a tensor describing the mapping and its negation by

$$\beta^{f}\left[Y_f = y_f, X_{[d]} = x_{[d]}\right] = \begin{cases} 1 & \text{if} \quad f\left[X_{[d]} = x_{[d]}\right] = y_f \\ 0 & \text{else} \end{cases} . \tag{5}$$

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

$$\beta^{f}\left[Y_f, X_{[d]}\right] = \epsilon_1\left[Y_f\right] \otimes f\left[X_{[d]}\right] + \epsilon_0\left[Y_f\right] \otimes \neg f\left[X_{[d]}\right] . \tag{6}$$

In our graphical notation this property is visualized by



We further provide a more detailed example in coordinate sensitive notation in the following.

Example 12 (Logical Negation and Conjunction) *The basis encodings of the negation $\neg : [2] \to [2]$ is the matrix*

$$\beta^{\neg}\left[Y_\neg, X\right] = \begin{array}{cc} & Y_\neg \\ & \begin{array}{cc} 0 & 1 \end{array} \\ X_0 \; \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \end{array}$$

The 2-ary conjunctions $\wedge : [2]\times[2] \to [2]$ is encoded by the order-3 tensor

$$\beta^{\wedge}\left[Y_\wedge, X_0, X_1\right] = \begin{array}{cc} Y_\wedge \; \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{array} \otimes \begin{array}{cc} & X_1 \\ & \begin{array}{cc} 0 & 1 \end{array} \\ X_0 \; \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \end{array} + \begin{array}{cc} Y_\wedge \; \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{array} \otimes \begin{array}{cc} & X_1 \\ & \begin{array}{cc} 0 & 1 \end{array} \\ X_0 \; \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \end{array} = \begin{array}{ccc} & X_1 & \\ & \begin{array}{cc} 0 & 1 \end{array} & \\ X_0 \; \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \\ & \underset{Y_\wedge}{0 \dashrightarrow 1} & \end{array}$$

Further, the 2-ary disjunction $\vee : [2]\times[2] \to [2]$ is encoded by the order-3 tensor

$$\beta^{\vee}\left[Y_\vee, X_0, X_1\right] = \begin{array}{ccc} & X_1 & \\ & \begin{array}{cc} 0 & 1 \end{array} & \\ X_0 \; \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \\ & \underset{Y_\vee}{0 \dashrightarrow 1} & \end{array}$$

Interpretation as CompAct Nets The propositional formula and its negation can be represented by that tensor by

$$f\left[X_{[d]}\right]=\left\langle \epsilon_1\left[Y_f\right],\beta^f\left[Y_f,X_{[d]}\right]\right\rangle_{\left[X_{[d]}\right]} \quad \text{and} \quad \neg f\left[X_{[d]}\right]=\left\langle \epsilon_0\left[Y_f\right],\beta^f\left[Y_f,X_{[d]}\right]\right\rangle_{\left[X_{[d]}\right]} .$$

Both f and $\neg f$ are thus Computation-Activation Networks to the statistic $\{f\}$ and the hard activation tensor $\epsilon_1\left[Y_f\right]$, respectively $\epsilon_0\left[Y_f\right]$. This representation of propositional formulas with respect to basis encoding thus 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.

4.2 Decomposition of Propositional Formulas

We now show, that the propositional formula allows for a decomposition into connective formulas, its basis encoding decomposes into the basis encodings of the connective formulas.

Lemma 24 *Let $f\left[X_{[d]}\right]$ be a composition of a p-ary connective formula $\circ$ and propositional formulas $f_l\left[X_{[d]}\right]$, where $l\in[p]$, i.e. for $x_{[d]}\in\bigtimes_{k\in[d]}[2]$ we have*

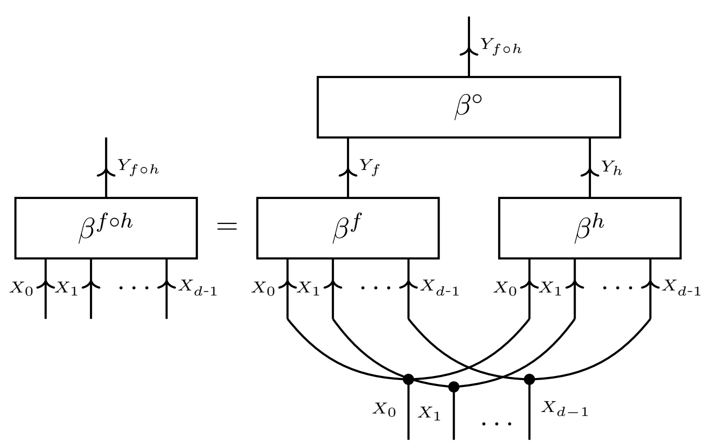
$$f\left[X_{[d]}=x_{[d]}\right]=\circ\left(f_0\left[X_{[d]}=x_{[d]}\right],\ldots,f_{p-1}\left[X_{[d]}=x_{[d]}\right]\right) .$$

Then we have

$$\beta^f\left[Y_f,X_{[d]}\right]=\left\langle\{\beta^\circ\left[Y_f,Y_{[p]}\right]\}\cup\{\beta^{f_l}\left[Y_l,X_{[d]}\right]\,:\,l\in[p]\}\right\rangle_{\left[Y_f,X_{[d]}\right]} .$$

Proof This can be shown on each index $x_{[d]}$. ■

For the composition of two propositional formulas $f\left[X_{[d]}\right]$ and $h\left[X_{[d]}\right]$ the composition by some binary connective is pictured by:



Let us now define a more generic syntactical decomposition of propositional formulas.

Definition 25 *A syntactical hypergraph is a directed acyclic hypergraph $\mathcal{G}=(\mathcal{V},\mathcal{E})$ such that*

- *each hyperedge* $e = (e^{\text{in}}, e^{\text{out}})$ *has exactly one outgoing node, i.e.* $\left|e^{\text{out}}\right| = 1$
- *each node* $v \in \mathcal{V}$ *carries a boolean variable* Y_v *and appears at most once as the outgoing node of a hyperedge*
- *each hyperedge* $(e^{\text{in}}, \{v\})$ *with* $e^{\text{in}} \neq \varnothing$ *is decorated by a propositional formula*

$$\circ_v \left[Y_{e^{\text{in}}}\right] \,:\, \bigtimes_{v \in e^{\text{in}}} [2] \to [2]$$

- *the node not appearing as an outgoing node are labeled by* $[d]$

We say that the syntactical hypergraph is single-rooted, if exactly one node $\tilde{v}$ *does not appear as an incoming node of a hyperedge. In this case this unique node is called the root node. We assign atomic formulas to the nodes* $[d]$ *and recursively assign to each further node* v *a node formula*

$$f_v\left[X_{[d]} = x_{[d]}\right] = \circ_v \left[\left[f_{\bar{v}}\left[X_{[d]} = x_{[d]}\right] \,:\, \bar{v} \in e^{\text{in}}\right]\right] \quad \forall x_{[d]} \in \bigtimes_{k \in [d]} [m_k]\,,$$

where e^{in} *are the incoming nodes in the unique hyperedge with outgoing nodes* $\{v\}$*. We call the formula* $f\left[X_{[d]}\right] \coloneqq f_{\tilde{v}}\left[X_{[d]}\right]$ *to the root note* $\tilde{v}$ *the syntactical composition of* $\mathcal{G}$ *and* $\mathcal{G}$ *is a syntactical decomposition of* f*.*

Theorem 26 *For any syntactical hypergraph* $\mathcal{G}$ *with composition* f *we have*

$$\begin{aligned} f\left[X_{[d]}\right] = \Big\langle & \left\{\beta^{\circ_v}\left[Y_v, Y_{e^{\text{in}}}\right] \,:\, (e^{\text{in}}, \{v\}) \in \mathcal{E}\right\} \cup \\ & \left\{\delta\left[Y_k, X_k\right] \,:\, k \in [d]\right\} \cup \left\{\epsilon_1\left[Y_{\tilde{v}}\right]\right\}\Big\rangle_{\left[X_{[d]}\right]}\,. \end{aligned}$$

Proof One can show this theorem by induction over the node formulas of the syntactical hypergraph, from the leafs to the root and iteratively applying Lem. 24. ■

Thus we have a tensor network representation of any propositional formula based on its syntactical decomposition, where the hypergraph of the syntactical decomposition equals the hypergraph of the representing tensor network.

4.3 Contractions to decide entailment

We have already seen that the contraction of a propositional formula counts its models. This allows to define entailment between two propositional formulas as follows.

Definition 27 (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*

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

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

Classically (see e.g. Russell and Norvig) entailment in propositional logics is defined as the the unsatisfiability of $\mathcal{KB} \wedge \neg f$. This is equivalent to Def. 27, since $\langle \mathcal{KB}, \neg f \rangle_{[\varnothing]} = 0$ is equivalent to $\langle \mathcal{KB} \wedge (\neg f) \rangle_{[\varnothing]} = 0$, which is the unsatisfiability of $\mathcal{KB} \wedge \neg f$.

Entailment is the central operation of "logical inference", i.e. deduce true statements from known statements. In the tensor network representation, these entailments can be decided by contracting the whole representing tensor with the statement, that needs to be checked.

Example 13 **($n^2\, x\, n^2$ Sudoku)** *We index the rows and the columns by tuples* $(r0, r1)$ *and* $(co, c1)$*, where* $r0, r1, c0, c1 \in [n]$*. The first index indicates the block and the second counts the row or column inside that block. For each* $r0, r1, c0, c1 \in [n]$ *and* $i \in [n^2]$ *we then define an atomic variable* $X_{r0,r1,c0,c1,i} \in \{0, 1\}$ *indicating whether in the row* $(r0, r1)$ *and column* $(co, c1)$ *the number* i *is written. The Sudoku rules then amount to the formula*

$$\mathcal{KB}^n := \left(\bigwedge_{r0,r1,c0,c1 \in [n]} \left(\bigoplus_{i \in [n^2]}^{(1)} X_{r0,r1,c0,c1,i} \right) \right) \wedge \left(\bigwedge_{r0,r1 \in [n], i \in [n^2]} \left(\bigoplus_{c0,c1 \in [n]}^{(1)} X_{r0,r1,c0,c1,i} \right) \right) \wedge \\ \left(\bigwedge_{c0,c1 \in [n], i \in [n^2]} \left(\bigoplus_{c0,c1 \in [n]}^{(1)} X_{r0,r1,c0,c1,i} \right) \right) \wedge \left(\bigwedge_{r0,c0 \in [n], i \in [n^2]} \left(\bigoplus_{r1,c1 \in [n]}^{(1)} X_{r0,r1,c0,c1,i} \right) \right) ,$$

where $\bigoplus^{(1)}$ *is the* n^2*-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* $(r_0, r_1, c_0, c_1, i) \notin E$ *whether the Sudoku rules and the initial board imply that the atomic variable* $X_{r0,r1,c0,c1,i}$ *(i.e. assignment to the board) is true*

$$\mathcal{KB}^n \wedge \left(\bigwedge_{(r_0,r_1,c_0,c_1,i) \in E} X_{r0,r1,c0,c1,i} \right) \models X_{r0,r1,c0,c1,i}$$

or false

$$\mathcal{KB} \wedge \left(\bigwedge_{(r_0,r_1,c_0,c_1,i) \in E} X_{r0,r1,c0,c1,i} \right) \models \neg X_{r0,r1,c0,c1,i} \, . \tag{7}$$

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* $(r0, r1)$ *and column* $(c0, c1)$*? 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* $(r_0, r_1, c_0, c_1, i) \notin E$*. Deciding which is equivalent to solving of the Sudoku.*

For example, let $n = 2$ and

$$E = \{(0,0,0,0,0), (0,0,1,0,2), (0,0,1,1,1), (0,1,0,1,1),\\ (1,0,1,0,3), (1,1,0,0,3), (1,1,0,1,2)\}\,.$$

We visualize this evidence by writing $i+1$ in a grid cell $(r0, r1, c0, c1)$ to indicate that $(r0, r1, c0, c1, i) \in E$:

1		3	2
	2		
		4	
4	3		

We will later demonstrate in Example 13 a solution algorithm to solve this instance.

4.4 Efficient Representation of Knowledge Bases

We now investigate the representation of knowledge bases, which are conjunctions

$$\mathcal{KB}\left[X_{[d]}\right] = \bigwedge_{l\in[p]} f_l\left[X_{[d]}\right]\,.$$

To show efficient repesentations we will use the following identities.

Lemma 28 (Computation Network Symmetries) *We have for the d-ary $\wedge$-connective (where $d \in \mathbb{N}$) and the unary $\neg$-connective that*

$$\left\langle \epsilon_1\left[Y\right], \beta^{\wedge}\left[Y, X_{[d]}\right]\right\rangle_{\left[X_{[d]}\right]} = \bigotimes_{k\in[d]} \epsilon_1\left[X_k\right] \quad \textit{and} \quad \left\langle \epsilon_1\left[Y\right], \beta^{\neg}\left[Y, X\right]\right\rangle_{[X]} = \epsilon_0\left[X\right]\,.$$

Proof Follows directly from the definitions of the basis encodings and the connectives. ■

We use this to decompose knowledge bases into their individual formulas as follows.

Theorem 29 *For any knowledge base $\mathcal{KB}\left[X_{[d]}\right] = \bigwedge_{l\in[p]} f_l\left[X_{[d]}\right]$ it holds that*

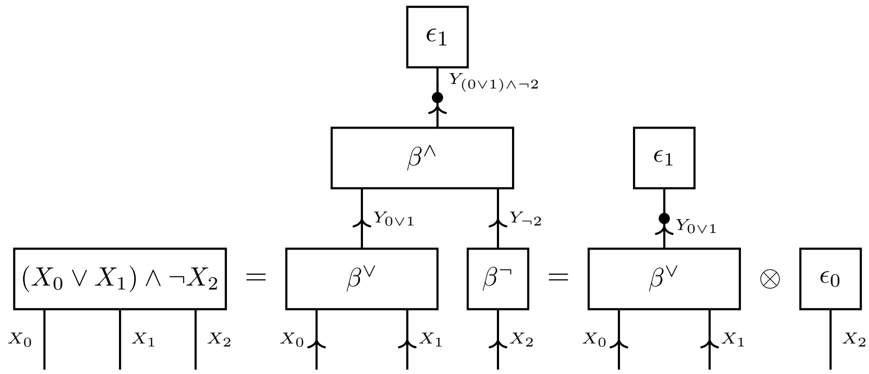
$$\mathcal{KB}\left[X_{[d]}\right] = \left\langle \left\{f_l\left[X_{[d]}\right] \,:\, l\in[p]\right\}\right\rangle_{\left[X_{[d]}\right]}\,.$$

Proof With Lem. 28 we have

$$\begin{aligned}
\mathcal{KB}\left[X_{[d]}\right] &= \left\langle \left\{\epsilon_1\left[Y_{\wedge}\right], \beta^{\wedge}\left[Y_{\wedge}, Y_{[p]}\right]\right\} \cup \left\{\beta^{f_l}\left[Y_l, X_{[d]}\right] \,:\, l\in[p]\right\}\right\rangle_{\left[X_{[d]}\right]}\\
&= \left\langle \bigcup_{l\in[p]} \left\{\epsilon_1\left[Y_l\right], \beta^{f_l}\left[Y_l, X_{[d]}\right] \,:\, l\in[p]\right\}\right\rangle_{\left[X_{[d]}\right]}\\
&= \left\langle \left\{f_l\left[X_{[d]}\right] \,:\, l\in[p]\right\}\right\rangle_{\left[X_{[d]}\right]}\,.
\end{aligned}$$

■

Example 14 (Computation Network Symmetries) *For the propositional formula $f\left[X_{[3]}\right] = (X_0 \vee X_1) \wedge \neg X_2$ (see Example 10), we can write the formula in terms of a Computation-Activation Network with activation tensor ϵ_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.*



4.5 Message-passing for Entailment

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.

Theorem 30 (Monotonocity of Propositional Logics) *If $\tilde{\mathcal{KB}} \subset \mathcal{KB}$ and $\tilde{\mathcal{KB}} \models f$ then also $\mathcal{KB} \models f$.*

Proof Since $\tilde{\mathcal{KB}} \models f$ it holds that $\left\langle \tilde{\mathcal{KB}}, \neg f \right\rangle_{[\varnothing]} = 0$ and thus $\left\langle \tilde{\mathcal{KB}}, \neg f \right\rangle_{\left[X_{[d]}\right]} = 0\left[X_{[d]}\right]$. Denoting by $\mathcal{KB}/\tilde{\mathcal{KB}}$ the conjunctions of formulas in $\mathcal{KB}$ not in $\tilde{\mathcal{KB}}$, we have

$$
\begin{aligned}
\left\langle \mathcal{KB}\left[X_{[d]}\right], \neg f\left[X_{[d]}\right] \right\rangle_{[\varnothing]} &= \left\langle \mathcal{KB}/\tilde{\mathcal{KB}}[X_{[d]}], \tilde{\mathcal{KB}}, \neg f\left[X_{[d]}\right] \right\rangle_{[\varnothing]} \\
&= \left\langle \mathcal{KB}/\tilde{\mathcal{KB}}[X_{[d]}], \left\langle \tilde{\mathcal{KB}}[X_{[d]}], \neg f[X_{[d]}] \right\rangle_{\left[X_{[d]}\right]} \right\rangle_{[\varnothing]} \\
&= \left\langle \mathcal{KB}/\tilde{\mathcal{KB}}[X_{[d]}], 0\left[X_{[d]}\right] \right\rangle_{[\varnothing]} \\
&= 0\,.
\end{aligned}
$$

■

To decide entailment, we can therefore investigate entailment on smaller parts of the knowledge base. This is sound by the above theorem, but not complete, since it can happen that no smaller part of the knowledge base entails the formula, but the whole knowledge base does.

We can futhermore add entailed formulas to the knowledge base without the latter, as we show next.

Theorem 31 (Invariance of adding Entailed Formulas) *If $\mathcal{KB} \models f$ then*

$$\mathcal{KB}\left[X_{[d]}\right] = \langle \mathcal{KB}, f \rangle_{\left[X_{[d]}\right]} .$$

Proof We use that $f[X_{[d]}] + \neg f[X_{[d]}] = \mathbb{I}\left[X_{[d]}\right]$ and thus

$$\begin{aligned}
\mathcal{KB}\left[X_{[d]}\right] &= \left\langle \mathcal{KB}[X_{[d]}], (f[X_{[d]}] + \neg f[X_{[d]}]) \right\rangle_{\left[X_{[d]}\right]} \\
&= \left\langle \mathcal{KB}[X_{[d]}], f[X_{[d]}] \right\rangle_{\left[X_{[d]}\right]} + \left\langle \mathcal{KB}[X_{[d]}], \neg f[X_{[d]}] \right\rangle_{\left[X_{[d]}\right]} \\
&= \left\langle \mathcal{KB}[X_{[d]}], f[X_{[d]}] \right\rangle_{\left[X_{[d]}\right]} .
\end{aligned}$$

■

One can understand this theorem 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.

This motivates an message-passing approach to decide entailment by iteratively adding entailed formulas to the knowledge base and checking entailment on smaller parts of the knowledge base. Such inference methods are nown e.g. as Constraint Propagation.

Example 15 (Constraint Propoagation for a $2^2\, x\, 2^2$ Sudoku) *We iteratively solve a Sudoku puzzle (see Example 13) by determining a possible value based on neighboring cells, rows and squares (using Thm. 30) and adding to our knowledge (using Thm. 31). For example, consider the following $r = 2$ Sudoku puzzle, where a first entailment step uses only the knowledge of the rules and the blue cells to determine the value 3 in the first square:*

1		3	2
	2		
		4	
4	3		

=

1		3	2
3	2		
		4	
4	3		

= … =

1	4	3	2
3	2	1	4
2	1	4	3
4	3	2	1

To illustrate the first reasoning step we make the following preliminary entailment steps applying Thm. 30:

- *From $X_{0,1,0,1,1}$ (i.e. the 2 in the cell $(0,1,0,1)$) and the Sudoku rule that at the cell $(0,1,0,1)$ exactly one number is assigned, we get*

$$\left(\bigoplus_{i \in [n^2]}^{(1)} X_{0,1,0,1,i} \right) \wedge X_{0,1,0,1,1} \models \neg X_{0,1,0,1,2} ,$$

That is, that the number 3 is not in the cell $(0,1,0,1)$.

- *From* $X_{0,0,1,0,2}$ *(i.e. the* 3 *in the cell* $(0,0,1,0)$*) and the Sudoku rule that at the row* $(0,0)$ *exactly one number is assigned, we get*

$$\left(\bigoplus_{c0,c1\in[n]}^{(1)} X_{0,0,c0,c1,2}\right) \land X_{0,0,1,0,2} \models \neg X_{0,0,0,0,2} \land \neg X_{0,0,0,1,2}\,,$$

  That is, that the number 3 *is neither in the cell* $(0,0,0,0)$ *not in* $(0,0,0,1)$.

We add these formulas to our knowledge base (justified by Thm. 31) and use the rule, that 3 *appears exactly once in the first square*

$$\left(\bigoplus_{r1,c1\in[n]}^{(1)} X_{0,r1,0,c1,2}\right) \land (\neg X_{0,1,0,1,2}) \land (\neg X_{0,0,0,0,2} \land \neg X_{0,0,0,1,2}) \models X_{0,1,0,0,2}\,.$$

That is, we conclude that the number 3 *must be in the cell* $(0,1,0,0)$*, which information is also included in the updated knowledge base for further reasoning steps.*

We now iteratively apply similar reasoning steps and store the entailed variables in E^{entailed}*, until we arrive at the right side of the above sketch.*

$$\mathcal{KB} \land \left(\bigwedge_{(r_0,r_1,c_0,c_1,i)\in E^{\text{start}}} X_{r0,r1,c0,c1,i}\right) \models \left(\bigwedge_{(r_0,r_1,c_0,c_1,i)\in E^{\text{entailed}}} X_{r0,r1,c0,c1,i}\right).$$

Since all Sudoku rules are satisfied in the final assignment and to each cell (r_0,r_1,c_0,c_1) *we found exactly one* $i\in[n^2]$ *such that* $(r_0,r_1,c_0,c_1,i)\in E^{\text{start}}\cup E^{\text{entailed}}$*, there is a unique solution of the puzzle and we conclude*

$$\begin{aligned}\mathcal{KB} \land &\left(\bigwedge_{(r_0,r_1,c_0,c_1,i)\in E^{\text{start}}} X_{r0,r1,c0,c1,i}\right)\\ &= \left(\bigwedge_{(r_0,r_1,c_0,c_1,i)\in E^{\text{start}}} X_{r0,r1,c0,c1,i}\right) \land \left(\bigwedge_{(r_0,r_1,c_0,c_1,i)\in E^{\text{entailed}}} X_{r0,r1,c0,c1,i}\right).\end{aligned}$$

5 Hybrid Logic Networks

Let us now exploit the common formulation of logical formulas and probabilistic models in CompAct Nets to define hybrid models, which combine both aspects. We call CompAct Nets in the special case of boolean statistics t Hybrid Logic Networks.

5.1 Parametrization

We now investigate hybrid logical and probabilistic models, which we want to call Hybrid Logic Networks (HLNs).

Definition 32 (Hybrid Logic Network (HLN)) *Given a boolean statistic t we call any element of $\Lambda^{t,\mathrm{EL}}$ a Hybrid Logic Network. The extended canonical parameter set to t is the set*

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

To each Hybrid Logic Network $\mathbb{P}^{t,(A,y_A,\theta)}\left[X_{[d]}\right]$ we find a tuple (A, y_A, θ) 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

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

where the activation core is

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

We notice that the parametrization by $\mathcal{P}_p$ is one-to-one for any non-vanishing elementary activation tensor up to a scalar factor. Given an arbitrary elementary activation tensor $\bigotimes_{l\in[p]} \xi^l[Y_l]$, we can always find a corresponding tuple in $\mathcal{P}_p$ by choosing[1]

$$A = \{l \,:\, \mathbb{I}_{\neq 0}\left(\xi^l[Y_l]\right) \neq \mathbb{I}[Y_l]\} \,,$$

further for all $l \in A$

$$y_l = \begin{cases} 0 & \text{if} \quad \mathbb{I}_{\neq 0}\left(\xi^l[Y_l]\right) = \epsilon_0[Y_l] \\ 1 & \text{if} \quad \mathbb{I}_{\neq 0}\left(\xi^l[Y_l]\right) = \epsilon_1[Y_l] \end{cases}$$

and a parameter vector $\theta[L] \in \mathbb{R}^p$ defined for all $l \in [p]$ as

$$\theta[L = l] = \begin{cases} 0 & \text{if} \quad l \in A \\ \ln\left[\frac{\xi^l[Y_l=1]}{\xi^l[Y_l=1]}\right] & \text{if} \quad l \notin A \,. \end{cases}$$

Then we have by construction that there is $\lambda > 0$ with

$$\bigotimes_{l\in[p]} \xi^l[Y_l] = \lambda \cdot \xi^{(A,y_A,\theta)}\left[Y_{[p]}\right] .$$

Let us demonstrate the utility of Hybrid Logic Networks with an example from accounting.

Example 16 (Hybrid Logic Network for a Toy Accounting Model) *Let us consider a system of three variables $A1$ Account 1 is booked, $A2$ Account 2 is booked, F a feature on an invoice. We respect two rules*

- *Exactly one account must be booked.*

1. Here $\mathbb{I}_{\neq 0}(\cdot)$ is the indicator of non-zero entries acting coordinatewise and $\mathbb{I}[Y_l]$ is the vector $[1, 1]^T$.

- *If feature* F *is present on the invoice, the account* A1 *is typically booked.*

We formalize this with the statistic

$$t = (X_{A1} \oplus X_{A2}, X_F \Rightarrow X_{A1})\,.$$

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

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

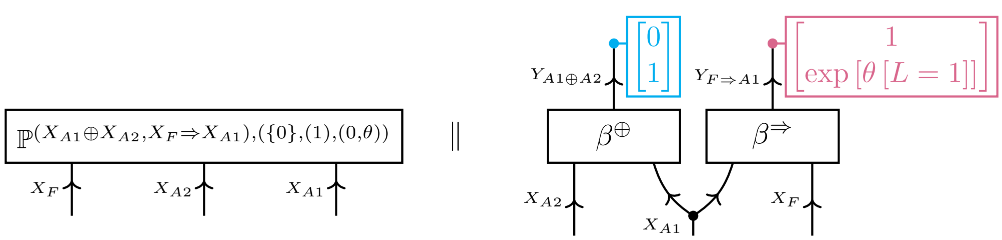
Since the first feature is hard, the "soft logic activation tensor" should be invariant under the first coordinate of the canonical parameter and we set $\theta[L=0] = 0$. We choose the soft parameters as $\theta[L] = [0, \theta[L=1]]^\intercal$ to achieve

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

The activation tensor of the hybrid network then has the from

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

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 $\mathbb{P}^{t,(A,y_A,\theta)}[X_{A1}, X_{A2}, X_F]$ of order 3. With $Y_{F \Rightarrow A_1} = 1$ for $F = 0$ and any A_1 it has the coordinates

$$\mathbb{P}^{(X_{A1}\oplus X_{A2}, X_F \Rightarrow X_{A1}),(\{0\},(1),(0,\theta))}[X_{A1}, X_{A2}, X_F] = \frac{1}{1+3\cdot\exp[\theta]} \quad X_{A1} \begin{array}{c} 0 \\ 1 \end{array} \underbrace{\overbrace{\begin{bmatrix} 0 & \exp[\theta] \\ \exp[\theta] & 0 \end{bmatrix}}^{X_{A2}:\ 0 \ \ 1}}_{X_F = 0} \ \underbrace{\begin{bmatrix} 0 & 1 \\ \exp[\theta] & 0 \end{bmatrix}}_{X_F = 1}$$

5.2 Parameter Estimation in Hybrid Logic Networks

Let us now briefly discuss how Hybrid Logic Networks can be trained on data based on likelihood maximization. Given a dataset $\left((x_0^j, \dots, x_{d-1}^j) \,:\, j \in [m]\right)$ consisting of m independent and identically distributed samples from an unknown distribution, we want to find a Hybrid Logic Network $\mathbb{P}^{t,(A,y_A,\theta)}\left[X_{[d]}\right]$ that maximizes the data likelihood

$$\mathcal{L}_D\left((A, y_A, \theta)\right) \coloneqq -\frac{1}{m} \sum_{i \in [n]} \ln\left[\mathbb{P}^{t,(A,y_A,\theta)}\left[X_{[d]} = x_{[d]}^j\right]\right] .$$

We notice that this is ∞ if and only if there is a data point $i \in [n]$ with

$$f^{t,(A,y_A)}\left[X_{[d]} = x_{[d]}^j\right] = 0 .$$

If this is not the case, we can rewrite the loss using the empirical mean vector $\mu_D[L] \in \mathbb{R}^p$, which is defined for $l \in [p]$ as

$$\mu_D[L = l] = \frac{1}{m} \sum_{i \in [n]} f_l\left[X_{[d]} = x_{[d]}^j\right] ,$$

by

$$\mathcal{L}_D\left((A, y_A, \theta)\right) = \left\langle \mu_D[L], \theta[L] \right\rangle_{[\varnothing]} - \ln\left[\left\langle \xi^{(A,y_A,\theta)}\left[Y_{[p]}\right], \beta^t\left[Y_{[p]}, X_{[d]}\right] \right\rangle_{[\varnothing]}\right] .$$

Since (A, y_A) influences only the second term the best hard parameters can be found by

$$A = \{l \,:\, \mu_D[L = l] \in \{0, 1\}\} \quad \text{and} \quad y_l = \mu_D[L = l] .$$

We further optimize the coordinates $l \in [p]/A$ of $\theta[L] \in \mathbb{R}^p$ alternatingly by the coordinate descent steps

$$\frac{\partial \mathcal{L}_D\left((A, y_A, \theta)\right)}{\partial \theta[L = l]} = 0 \Leftrightarrow \theta[L = l] = \ln\left[\frac{\mu[L = l]}{(1 - \mu[L = l])} \cdot \frac{\tau[Y_l = 0]}{\tau[Y_l = 1]}\right] .$$

where

$$\tau[Y_l] = \left\langle \{\beta^{f_l} \,:\, l \in [p]\} \cup \{\alpha^{\tilde{l},\theta} \,:\, \tilde{l} \in [p], \tilde{l} \neq l\} \cup \{\nu\} \right\rangle_{[Y_l]} .$$

Based on an interpretation of the coordinate descent steps as matching steps for the mean parameters or moments to f_l, we call this method alternating moment matching for Hybrid Logic Networks and provide pseudocode for it it in Algorithm 1. We notice, that during the coordinate descent steps the computation of the marginal probability of the variable Y_l with respect to the current network parameters is required. This is the computational bottleneck of the algorithm and can be approached by various approximate inference methods, e.g., variational inference (see for example the CAMEL method Ganapathi et al.).

It can be shown, that the algorithm converges if and only if there is a Hybrid Logic Network matching the empirical moments of the data. For more details we refer to (Goessmann, 2025, Chapter 9).

Algorithm 1 Alternating Moment Matching for Hybrid Logic Networks

Require: Mean parameter $\mu_D[L]$

Ensure: Canonical parameter $\theta[L]$, such that $\mathbb{P}^{(t,\theta,\nu)}$ is the (approximative) moment projection of $\mathbb{P}^D$ onto $\Lambda^{\mathcal{F},\mathrm{EL}}$

Set

$$A = \Big\{ l \,:\, l \in [p],\, \mu[L=l] \in \{0,1\} \Big\}$$

and a tuple y_A with $y_l = \mu[L=l]$ for $l \in A$.

Set $\theta[L] = 0[L]$

while Convergence criterion is not met **do**

  for all $l \in [p]/A$ **do**

    Compute

$$\tau[Y_l] = \Big\langle \{\beta^{f_l} \,:\, l \in [p]\} \cup \{\alpha^{\tilde{l},\theta} \,:\, \tilde{l} \in [p],\, \tilde{l} \neq l\} \cup \{\nu\} \Big\rangle_{[Y_l]}$$

    Set

$$\theta[L=l] = \ln\left[\frac{\mu[L=l]}{(1-\mu[L=l])} \cdot \frac{\tau[Y_l=0]}{\tau[Y_l=1]}\right]$$

  end for

end while

return $(A, y_A, \theta[L])$

Example 17 (Continuation of Example 16) *Let us recall the statistic of Example 16 and consider a dataset of $m = 20$ states summarized in the frequency table:*

Frequency in Dataset	x_{A1}	x_{A2}	x_F
0	0	0	0
0	0	0	1
7	0	1	0
2	0	1	1
1	1	0	0
10	1	0	1
0	1	1	0
0	1	1	1

We then have for the satisfaction rates of $f_0 = X_{A1} \oplus X_{A2}$ and $f_1 = X_F \Rightarrow X_{A1}$

$$\mu_D[L=0] = \frac{20}{20} = 1 \quad \textit{and} \quad \mu_D[L=1] = \frac{7+1+10}{20} = 0.9\,.$$

Then Algorithm 1 yields with a reasonable convergence criterion choice (such as finite iterations or convergence of $\theta[L]$)

$$A = \{0\} \quad , \quad y_A = 1 \quad \textit{and} \quad \theta[L] = \begin{bmatrix} 0 \\ \ln\left[\left(\frac{0.25}{0.75}\right)\cdot\left(\frac{0.9}{0.1}\right)\right] \end{bmatrix} = \begin{bmatrix} 0 \\ \ln[3] \end{bmatrix} \approx \begin{bmatrix} 0 \\ 1.098612 \end{bmatrix}.$$

5.3 Entailment by Hybrid Logic Networks

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 33 ((Goessmann, 2025, Theorem 8.12)) *Unclear: Would need to add probabilistic entailment.. Let $\mathbb{P}^{t,(A,y_A,\theta)}\left[X_{[d]}\right]$ be a Hybrid Logic Network. Given a query formula g, we have that $\mathbb{P}^{t,(A,y_A,\theta)}\left[X_{[d]}\right] \models g$ if and only if*

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

where

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

Proof Could be done based on support. ■

Example 18 *Entailment for Accounting Logic Bad example: Minterms are only entailed by themselves. To check entailment of the query formula defined by the minterm*

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

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

$$f^{t,(A,y_A)}[X_{A_1}, X_{A_2}, X_F] = X_{A1} \oplus X_{A2} \otimes \mathbb{I}$$

with g arriving at

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

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[2]. 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 12 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 12.

```
'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}
    )
```

As in example 12, the CAN then has the following form.

The other representation in example 12 can also be implemented. Both architectures can be found the linked notebook[3]. 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.

2. https://github.com/EnexaProject/enexa-tensor-reasoning/tree/version1
3. https://colab.research.google.com/drive/14knFuMJHI683DAmUgJ-G10MQFueoXR6q#scrollTo=vr0YBViZhX2S

```
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 frequenctly applied schemes are particle-based inference schemes such as Gibbs sampling.

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

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.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition edition. ISBN 978-0-471-24195-9.

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.

Varun Ganapathi, David Vickrey, John Duchi, and Daphne Koller. Constrained approximate maximum entropy learning of markov random fields. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, UAI'08, pages 196–203. AUAI Press. ISBN 978-0-9749039-4-1.

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 L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. 6(1): 164–189. ISSN 1467-9590. doi: https://doi.org/10.1002/sapm192761164.

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`.

Kevin P. Murphy. *Probabilistic Machine Learning: An Introduction*. The MIT Press. ISBN 978-0-262-04682-4.

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/iay009.

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.

Appendix A. Proof of the Factorization Theorems

Let us now provide proofs for the factorization theorems stated in Sect. 3. These proofs are classically known (see e.g. Koller and Friedman for Hammersley-Clifford and **?** for Fisher-Neyman). We here provide them in our tensor networks notation and for hypergraphs for completeness.

A.1 Hammersley-Clifford

Different to the original statement (see Clifford and Hammersley), we here proof the analogous statement for hypergraphs, where we have to demand the property of clique-capturing defined in Def. 17. We start with showing the following Lemmata to be exploited in the proof.

Lemma 34 *Let $\tau[X_{\mathcal{V}}]$ be a positive tensor and $y_{\mathcal{V}}$ an arbitrary index. Then we have*

$$\tau[X_{\mathcal{V}}] = \left\langle \left(\langle \tau \rangle_{[X_{\mathcal{V}/\mathcal{W}}, X_{\mathcal{W}} = y_{\mathcal{W}}]} \right)^{(-1)^{|\mathcal{U}|-|\mathcal{W}|}} : \mathcal{W} \subset \mathcal{U} \subset \mathcal{V} \right\rangle_{[X_{\mathcal{V}}]},$$

where the exponentiation is performed coordinatewise and positivity of τ ensures the well-definedness.

Proof It suffices to show, that for an arbitrary index $x_{\mathcal{V}}$ be an arbitrary index we have

$$\tau\left[X_{\mathcal{V}}=x_{\mathcal{V}}\right]=\prod_{\mathcal{U}\subset\mathcal{V}}\prod_{\mathcal{W}\subset\mathcal{U}}\left(\langle\tau\rangle_{\left[X_{\mathcal{V}/\mathcal{W}}=x_{\mathcal{V}/\mathcal{W}},X_{\mathcal{W}}=y_{\mathcal{W}}\right]}\right)^{(-1)^{|\mathcal{U}|-|\mathcal{W}|}}.$$

We do this by applying a logarithm on the right hand side and grouping the terms by $\mathcal{W}$ as

$$\begin{aligned}
&\ln\left[\prod_{\mathcal{U}\subset\mathcal{V}}\prod_{\mathcal{W}\subset\mathcal{U}}\langle\tau\rangle_{\left[X_{\mathcal{V}/\mathcal{W}}=x_{\mathcal{V}/\mathcal{W}},X_{\mathcal{W}}=y_{\mathcal{W}}\right]}\big)^{(-1)^{|\mathcal{U}|-|\mathcal{W}|}}\right]\\
&=\sum_{\mathcal{W}\subset\mathcal{V}}\ln\left[\langle\tau\rangle_{\left[X_{\mathcal{V}/\mathcal{W}}=x_{\mathcal{V}/\mathcal{W}},X_{\mathcal{W}}=y_{\mathcal{W}}\right]}\right]\left(\sum_{\mathcal{U}\subset\mathcal{V}\,:\,\mathcal{W}\subset\mathcal{U}}(-1)^{|\mathcal{U}|-|\mathcal{W}|}\right)\\
&=\sum_{\mathcal{W}\subset\mathcal{V}}\ln\left[\langle\tau\rangle_{\left[X_{\mathcal{V}/\mathcal{W}}=x_{\mathcal{V}/\mathcal{W}},X_{\mathcal{W}}=y_{\mathcal{W}}\right]}\right]\left(\sum_{i\in[|\mathcal{V}|-|\mathcal{W}|]}(-1)^{i}\binom{|\mathcal{V}|-|\mathcal{W}|}{i}\right)
\end{aligned}$$

Now, by the generic binomial theorem we have that for $n\in\mathbb{N},n\neq 0$

$$0=(1-1)^{n}=\sum_{i\in[n]}(-1)^{i}\binom{n}{i}.$$

Therefore, the summands for $\mathcal{W}\neq\mathcal{V}$ vanish and we have

$$\begin{aligned}
&\ln\left[\prod_{\mathcal{U}\subset\mathcal{V}}\prod_{\mathcal{W}\subset\mathcal{U}}\left(\langle\tau\rangle_{\left[X_{\mathcal{V}/\mathcal{W}}=x_{\mathcal{V}/\mathcal{W}},X_{\mathcal{W}}=y_{\mathcal{W}}\right]}\right)^{(-1)^{|\mathcal{U}|-|\mathcal{W}|}}\right]\\
&=\ln\left[\tau\left[X_{\mathcal{V}}=x_{\mathcal{V}}\right]\right]\left(\sum_{i\in[0]}(-1)^{i}\binom{0}{i}\right)\\
&=\ln\left[\tau\left[X_{\mathcal{V}}=x_{\mathcal{V}}\right]\right].
\end{aligned}$$

Applying the exponential function on both sides establishes the claim. ∎

Lemma 35 *Let τ be a positive tensor, $\mathcal{U}\subset\mathcal{V}$ and arbitrary subset and $x_{\mathcal{U}}$ an arbitrary index. When there are $a,b\in\mathcal{U}$, such that*

$$\langle\tau\rangle_{\left[X_a,X_b|X_{\mathcal{V}/\{a,b\}}\right]}=\left\langle\langle\tau\rangle_{\left[X_a|X_{\mathcal{V}/\{a,b\}}\right]},\langle\tau\rangle_{\left[X_b|X_{\mathcal{V}/\{a,b\}}\right]}\right\rangle_{[X_{\mathcal{U}}]}$$

then

$$\prod_{\mathcal{W}\subset\mathcal{U}}\left(\langle\tau\rangle_{\left[X_{\mathcal{V}/\mathcal{W}}=x_{\mathcal{V}/\mathcal{W}},X_{\mathcal{W}}=y_{\mathcal{W}}\right]}\right)^{(-1)^{|\mathcal{U}|-|\mathcal{W}|}}=1\,.$$

Proof We abbreviate

$$Z_{\mathcal{W}}=\langle\tau\rangle_{\left[X_{\mathcal{V}/\mathcal{W}}=x_{\mathcal{V}/\mathcal{W}},X_{\mathcal{W}}=y_{\mathcal{W}}\right]}.$$

By reorganizing the sum over $\mathcal{W} \subset \mathcal{U}$ into $\mathcal{W} \subset \mathcal{U}/a \cup b$ we have

$$\prod_{\mathcal{W}\subset\mathcal{U}} (Z_{\mathcal{W}})^{(-1)^{|\mathcal{U}|-|\mathcal{W}|}} = \prod_{\mathcal{W}\subset\mathcal{U}/\{a,b\}} \left(\frac{Z_{\mathcal{W}} \cdot Z_{\mathcal{W}\cup\{a,b\}}}{Z_{\mathcal{W}\cup\{a\}} \cdot Z_{\mathcal{W}\cup\{b\}}}\right)^{(-1)^{|\mathcal{U}|-|\mathcal{W}|}} . \tag{8}$$

From the independence assumption it follows that for any index x

$$\begin{aligned}
&\langle\tau\rangle\left[X_a=x_a|X_{\mathcal{V}/\mathcal{W}\cup\{a,b\}}=x_{\mathcal{V}/\mathcal{W}\cup\{a,b\}},X_{\mathcal{W}}=y_{\mathcal{W}},X_b=x_b\right] \\
&\quad = \langle\tau\rangle\left[X_a=x_a|X_{\mathcal{V}/\mathcal{W}\cup\{a,b\}}=x_{\mathcal{V}/\mathcal{W}\cup\{a,b\}},X_{\mathcal{W}}=y_{\mathcal{W}}\right] \\
&\quad = \langle\tau\rangle\left[X_a=x_a|X_{\mathcal{V}/\mathcal{W}\cup\{a,b\}}=x_{\mathcal{V}/\mathcal{W}\cup\{a,b\}},X_{\mathcal{W}}=y_{\mathcal{W}},X_b=y_b\right]
\end{aligned}$$

Applying this in each squares bracket term of (8) we get

$$\begin{aligned}
\frac{Z_{\mathcal{W}}}{Z_{\mathcal{W}\cup\{a\}}} &= \frac{\langle\tau\rangle\left[X_a=x_a|X_{\mathcal{V}/\mathcal{W}\cup\{a,b\}}=x_{\mathcal{V}/\mathcal{W}\cup\{a,b\}},X_{\mathcal{W}}=y_{\mathcal{W}},X_b=x_b\right]}{\langle\tau\rangle\left[X_a=y_a|X_{\mathcal{V}/\mathcal{W}\cup\{a,b\}}=x_{\mathcal{V}/\mathcal{W}\cup\{a,b\}},X_{\mathcal{W}}=y_{\mathcal{W}},X_b=x_b\right]} \\
&= \frac{\langle\tau\rangle\left[X_a=x_a|X_{\mathcal{V}/\mathcal{W}\cup\{a,b\}}=x_{\mathcal{V}/\mathcal{W}\cup\{a,b\}},X_{\mathcal{W}}=y_{\mathcal{W}},X_b=y_b\right]}{\langle\tau\rangle\left[X_a=y_a|X_{\mathcal{V}/\mathcal{W}\cup\{a,b\}}=x_{\mathcal{V}/\mathcal{W}\cup\{a,b\}},X_{\mathcal{W}}=y_{\mathcal{W}},X_b=y_b\right]} \\
&= \frac{Z_{\mathcal{W}\cup\{b\}}}{Z_{\mathcal{W}\cup\{a,b\}}} .
\end{aligned}$$

Thus, each factor in (8) is trivial, which establishes the claim. ■

We are finally ready to prove the Hammersley-Clifford Thm. 18 based on the Lemmata above.

Proof [Proof of Thm. 18] By Lem. 34 we have for any index $x_{\mathcal{V}}$

$$\mathbb{P}\left[X_{\mathcal{V}} = x_{\mathcal{V}}\right] = \prod_{\mathcal{U}\subset\mathcal{V}} \prod_{\mathcal{W}\subset\mathcal{U}} \left(\mathbb{P}\left[X_{\mathcal{W}} = x_{\mathcal{W}}, X_{\mathcal{V}/\mathcal{W}} = y_{\mathcal{V}/\mathcal{W}}\right]\right)^{(-1)^{|\mathcal{U}|-|\mathcal{W}|}} .$$

Using the assumption of Thm. 18 we find for any subset $\mathcal{U} \subset \mathcal{V}$, which is not contained in a hyperedge, $a, b \in \mathcal{U}$ such that X_a is independendent on X_b conditioned on $X_{\mathcal{U}/\{a,b\}}$. If no such nodes $a, b \in \mathcal{U}$ exists, $\mathcal{U}$ would be contained in a hyperedge, since the hypergraph is assumed to be clique-capturing. By Lem. 35 we then have

$$\prod_{\mathcal{W}\subset\mathcal{U}} \left(\mathbb{P}\left[X_{\mathcal{W}} = x_{\mathcal{W}}, X_{\mathcal{V}/\mathcal{W}} = y_{\mathcal{V}/\mathcal{W}}\right]\right)^{(-1)^{|\mathcal{U}|-|\mathcal{W}|}} = 1 .$$

We label by a function

$$\alpha : \{\mathcal{U} : \exists e \in \mathcal{E} : \mathcal{U} \subset e\} \to \mathcal{E}$$

the remaining node subsets by a hyperedge containing the subset. We build the tensor

$$\tau^e\left[X_e\right] = \prod_{\mathcal{U}\,:\,\alpha(\mathcal{U})=e} \prod_{\mathcal{W}\subset\mathcal{U}} \left(\mathbb{P}\left[X_{\mathcal{W}} = x_{\mathcal{W}}, X_{\mathcal{V}/\mathcal{W}} = y_{\mathcal{V}/\mathcal{W}}\right]\right)^{(-1)^{|\mathcal{U}|-|\mathcal{W}|}} .$$

and get, that

$$\begin{aligned}\mathbb{P}[X_{\mathcal{V}}] &= \langle\{\tau^e[X_e] \,:\, e \in \mathcal{E}\}\rangle_{[X_{\mathcal{V}}]}\\ &= \langle\{\tau^e[X_e] \,:\, e \in \mathcal{E}\}\rangle_{[X_v|\varnothing]}\,.\end{aligned}$$

We have thus constructed a Markov Network with trivial partition function, which contraction coincides with the probability distribution. ■

A.2 Fisher-Neyman

Since sufficient statistics are sometimes introduced based on the data processing inequality (see e.g. Cover and Thomas), we also show that also that definition is equivalent to the factorization of the family.

Theorem 36 (Factorization Theorem of Fisher and Neyman) *Let $\mathbb{P}$ be a joint distribution of variables Z, X with values $\mathrm{val}(Z)$, $\mathrm{val}(X)$ and let $t(x)$ be a statistic. The following are equivalent:*

i) The Data Processing Inequality holds straight, i.e.

$$I(Z;X) = I(Z;Y_t)$$

ii) $Z \to Y_t \to X$ is a Markov Chain, i.e.

$$(Z \perp X) \mid Y_t$$

iii) There are tensors $\xi[Y_t, Z]$ and $\nu[X]$ such that

$$\mathbb{P}[Z=z, X=x] = \xi[Y_t = t(x), Z=z] \cdot \nu[X=x]\,.$$

Proof $i) \Leftrightarrow ii)$: We have always

$$I(Z;X) = I(Z;(X,Y_t)) = I(Z;Y_t) + I(Z;X|Y_t)$$

and thus if and only if $i)$ holds

$$I(Z;X|Y_t) = 0\,.$$

Using the KL-divergence characterization of the mutual information, this is equal to

$$\mathbb{P}\big[Z, X\big|Y_t\big] = \big\langle \mathbb{P}\big[Z\big|Y_t\big], \mathbb{P}\big[X\big|Y_t\big]\big\rangle_{[Z,X,Y_t]}\,.$$

This is equivalent to the conditional independence statement $ii)$.

$ii) \Rightarrow iii)$: Let us assume $ii)$. For almost all $z \in \mathrm{val}(Z)$ and $x \in \mathrm{val}(X)$ we then have

$$\begin{aligned}\mathbb{P}\big[Z=z\big|X=x\big] &= \mathbb{P}\big[Z=z\big|X=x, Y_t = t(x)\big]\\ &= \mathbb{P}\big[Z=z\big|Y_t = t(x)\big]\end{aligned}$$

Here we used that Y_t has a deterministic dependence on X. There is thus a tensor ξ such that for all $z \in \mathrm{val}(Z)$ and $x \in \mathrm{val}(X)$

$$\xi\left[Y_t = t(x), Z = z\right] = \mathbb{P}\left[Z = z \middle| X = x\right] .$$

We further define a tensor $\nu[X] = \mathbb{P}[X]$ and get

$$\begin{aligned}\mathbb{P}\left[Z = z, X = x\right] &= \mathbb{P}\left[X = x\right] \cdot \mathbb{P}\left[Z = z \middle| X = x\right] \\ &= \xi\left[Y_t = t(x), Z = z\right] \cdot \nu\left[X = x\right] .\end{aligned}$$

iii) $\Rightarrow$ *ii)*: When assuming *iii)* we have for all $(x, z) \in \mathrm{val}(Z) \times \mathrm{val}(X)$

$$\begin{aligned}\mathbb{P}\left[Z = z \middle| X = x\right] &= \left\langle \xi\left[Y_t, Z\right], \beta^t\left[Y_t, X\right], \nu\left[X\right] \right\rangle_{[Z=z|X=x]} \\ &= \left\langle \xi\left[Y_t, Z\right], \beta^t\left[Y_t, X = x\right], \nu\left[X = x\right] \right\rangle_{[Z=z|\varnothing]} \\ &= \left\langle \xi\left[Y_t, Z\right], \epsilon_{t(x)}\left[Y_t\right] \right\rangle_{[Z=z|\varnothing]} \\ &= \mathbb{P}\left[Z = z \middle| Y_t = t(x)\right] .\end{aligned}$$

We further have at almost all $y_t \in \mathrm{val}(Y_t)$, $z \in \mathrm{val}(Z)$ and $x \in \mathrm{val}(X)$ that $y_t = t(x)$ and

$$\mathbb{P}\left[Z = z \middle| X = x, Y_t = y_t\right] = \mathbb{P}\left[Z = z \middle| X = x\right]$$

and with the above at thus at almost all such pairs

$$\mathbb{P}\left[Z = z \middle| X = x, Y_t = y_t\right] = \mathbb{P}\left[Z = z \middle| Y_t = y_t\right] .$$

This is equivalent to *ii)*. ■

Thm. 20 follows from Thm. 36 by the equivalence of *ii)* and *iii)*.