

# CompActNets - A Tensor-Network architecture for Neuro-Symbolic AI

**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

This work introduces the novel Computation-Activation Networks (CompActNets), a tensor-network architecture that can represent probabilistic modeling and logical reasoning within a single mathematical framework. Propositional formulas and exponential-family distributions are unified in the resulting Hybrid Logic Networks. Computation or features, which are encoded by basis representations of sufficient statistics or logical formulas, are separated from activation, which assigns probabilistic or boolean values to these features. This separation yields sparse and interpretable representations that subsume graphical models, exponential-family factorizations, and logical inference as special cases. Probabilistic and logical queries reduce to tensor contractions, providing a common operational substrate for both reasoning paradigms. Altogether, CompActNets offer an expressive and intrinsically explainable foundation for neuro-symbolic AI, enabling exact reasoning alongside efficient implementation for learning tasks.

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

## 1 Introduction

Modern artificial intelligence is dominated by large-scale neural models that excel at various tasks but mostly remain black-boxes. In contrast, reliability and explainability are two main concerns when integrating these architecture into safety-critical processes. For these goals classical symbolic approaches offer explicit logical structures and human-readable inference but cannot handle uncertainty or scale to complex real-world data. Probabilistic models on the other hand improved uncertainty handling 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 Lipton (2018); Barredo Arrieta et al. (2020). The proposed framework encodes symbolic relations that remain directly readable. Explainability is thus not added after training but built into the architecture itself.

The logical tradition of artificial intelligence is motivated by the resemblance of human thought in logics McCarthy (1959). 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. (2016); Getoor and Taskar (2019) 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 (2022); Sarker et al. (2022); Colelough and Regli (2024), with close relations to statistical relational AI Marra et al. (2024). Neuro-Symbolic AI focuses on the unification of the neural and the symbolic paradigm Garcez et al. (2019), where early approaches are Towell and Shavlik (1994); Avila Garcez and Zaverucha (1999). 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 (deep) neural networks, whose size and encoding structure prevents human understanding of the inference process, Neuro-Symbolic AI aims at a re-implementation of the symbolic paradigm into such architectures. It hence provides systematic means to formalize the encoded data and knowledge as an abstraction in an explicit form.

The unification of symbolic and probabilistic approaches to interpretable model architectures has been a long-standing aim. Probabilistic graphical models Pearl (1988); Koller and Friedman (2009) are means to encode variable independences in graphs and are specific instances of exponential families Wainwright and Jordan (2008). Markov Logic Networks Richardson and Domingos (2006) are specific instances of exponential families where also the dependencies are explicitly encoded in logical formulas. Further approaches treat uncertainties as generalized truth values Bach et al. (2017).

Tensor Networks have recently gained interest as a unifying language for AI, framed by Logical Tensor Networks Badreddine et al. (2022) and Tensor Logic Domingos (2025). Different to the approaches therein we do not require non-linear transforms of tensors. Further, the MeLoCoToN approach Ali (2025) applies tensor network architectures similar to CompActNets in combinatorical optimization problems.

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 (2012). 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 (1993), 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. (2025b,a). 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 Orús (2019); Berezutskii et al. (2025).

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. (2020). Furthermore, they exhibit properties helpful for handling these high-dimensional spaces, such as restricted isometry properties Goessmann (2021). 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. 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.

The recently developed *tnreason* framework Goessmann (2025) proposes tensor networks as a unifying mathematical foundation for reasoning and probabilistic inference. Tensor networks factor complex systems into interconnected logical and probabilistic components. In this unifying mathematical framework, logical formulas corresponding to boolean tensors and probabilistic distributions corresponding to non-negative real tensors are combined. Both can be manipulated through the same algebra of tensor contraction. This abstraction eliminates the traditional divide between symbolic and numerical representations: logical inference and probabilistic computations become different instances of the same underlying operation on structured tensors. The *Computation-Activation Networks (CompActNets)* organize 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.

## 1.1 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 sub-

sequent 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 coding examples in section 6 that illustrate the expressive power and interpretability of this unified tensor-based reasoning approach.

## 2 Notation and Basic Concepts

Novelty here: Defining tensor networks on hypergraphs, to ease the relation with graphical models (avoid graph duality).

We in this section introduce the tensor network formalism and the basic architecture. First, the considered variables and tensors are explained. Followed by the explanation of tensor calculations, mainly tensor products and contractions. Finally, the main architecture, the *Computation-Activation Network*, is defined.

### 2.1 Tensors

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

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

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

for index tuples

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

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

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

which is a linear space, enriched with the operations of coordinate wise summation and scalar multiplication. We call a tensor  $\tau[X_{[d]}]$  boolean, when  $\text{im}(\tau) \subset [2]$ , i.e. all coordinates are either 0 or 1. We further abbreviate sets as  $[d] = \{0, \dots, d-1\}$ , tuples of state indices by  $x_{[d]} = x_0, \dots, x_{d-1}$  and tuples of variables by  $X_{[d]} = X_0, \dots, X_{d-1}$ .

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)** *Maybe directly do the Dirac tensor (special case of  $d = 1$  is the trivial tensor).* The trivial tensor

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

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

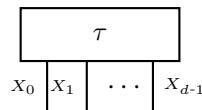
$$\mathbb{I}[X_{[d]} = x_{[d]}] = 1.$$

## 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 (2011) and dates back at least to the work Penrose (1987)). 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



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

**Definition 2 (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

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



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

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  $\mathcal{T}^{\mathcal{G}}$ .

As examples we now present the CP and the TT formats in our hypergraph notation.

**Example 2 (The CP format)** The Candecomp-Parafac (CP (Hitchcock (1927)) tensor format corresponds in our notation with a hypergraph (see Figure 1)

- 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  $\tau^{e_k} [X_k, I]$ .

**Example 3 (The TT format)** The Tensor-Train (TT) (see Oseledets (2011)) format corresponds in our notation with a hypergraph (see Figure 2)

- Nodes by  $X_{[d]}$  and hidden variables  $I_{[d-1]}$ , each decorated by a dimension  $m_{[d]}$  and  $n_{[d-1]}$
- Edges by

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

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

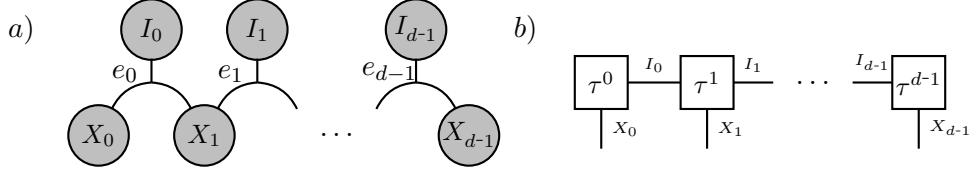


Figure 2: Hypergraph to a TT format. a) Node-centric design. b) Corresponding tensor-network on the edges of the hypergraph.

### 2.3 Generic Contractions

Let us now exploit the developed graphical representations to define contractions of tensor networks.

Contractions of Tensor Networks  $\tau^G$  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.

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

$$\langle \tau^G \rangle_{[X_{\mathcal{U}}]} \in \bigotimes_{v \in \mathcal{U}} \mathbb{R}^{m_v} \quad (1)$$

defined coordinatewise by the sum

$$\langle \tau^G \rangle_{[X_{\mathcal{U}}=x_{\mathcal{U}}]} = \sum_{x_{\mathcal{V}/\mathcal{U}} \in \times_{v \in \mathcal{V}/\mathcal{U}} [m_v]} \left( \prod_{e \in \mathcal{E}} \tau^e [X_e = x_e] \right). \quad (2)$$

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 ??).

**Example 4 (Tensor Product)** 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. Let there be two tensors

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

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

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

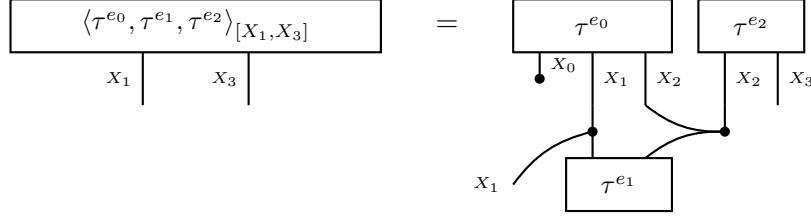


Figure 3: Graphical depiction of a tensor network contraction with the open variables  $X_1, X_3$ . Open variables are depicted by those without a dot at the end of the line.

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

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

## 2.4 Normalizations

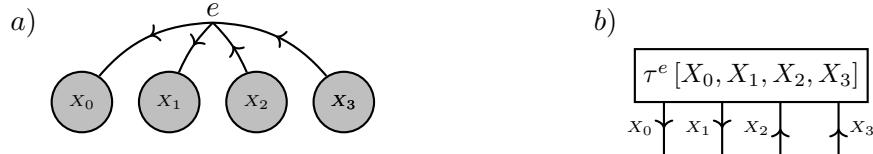
**Definition 4** The normalization of a tensor  $\tau [X_{\mathcal{V}}]$  on incoming nodes  $\mathcal{V}^{\text{in}} \subset \mathcal{V}$  and outgoing nodes  $\mathcal{V}^{\text{out}} \subset \mathcal{V}/\mathcal{V}^{\text{in}}$  is the tensor  $\langle \tau [X_{\mathcal{V}}] \rangle_{[X_{\mathcal{V}^{\text{out}}} | X_{\mathcal{V}^{\text{in}}}]}$  defined for  $x_{\mathcal{V}^{\text{in}}}$  as

$$\langle \tau [X_{\mathcal{V}}] \rangle_{[X_{\mathcal{V}^{\text{out}}} | X_{\mathcal{V}^{\text{in}}} = x_{\mathcal{V}^{\text{in}}}]} = \begin{cases} \frac{\langle \tau \rangle_{[X_{\mathcal{V}^{\text{out}}}, X_{\mathcal{V}^{\text{in}}} = x_{\mathcal{V}^{\text{in}}}]}}{\langle \tau \rangle_{[X_{\mathcal{V}^{\text{in}}} = x_{\mathcal{V}^{\text{in}}}]}} & \text{if } \langle \tau \rangle_{[X_{\mathcal{V}^{\text{in}}} = x_{\mathcal{V}^{\text{in}}}]} \neq 0 \\ \frac{1}{\prod_{v \in \mathcal{V}^{\text{out}}} m_v} \mathbb{I} [X_{\mathcal{V}^{\text{out}}}] & \text{else} \end{cases}.$$

We say that  $\tau [X_{\mathcal{V}}]$  is normalized with incoming nodes  $e^{\text{in}} \subset \mathcal{V}$ , if

$$\tau [X_{\mathcal{V}}] = \langle \tau [X_{\mathcal{V}}] \rangle_{[X_{\mathcal{V}/\mathcal{V}^{\text{in}}} | X_{\mathcal{V}^{\text{in}}}]}.$$

Directionality thus represents constraints on the structure of tensors, namely that the sum over outgoing trivializes the tensor. In our graphical tensor notation, we depict normalized tensors by directed hyperedges (a), which are decorated by directed tensors (b), for example:



## 2.5 Function encoding and Computation-Activation Networks

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 5 (One-hot encodings)** *Do it for factored directly* Given an atomic system described by the categorical variable  $X$ , we define for each  $x \in [m]$  the basis vector  $\epsilon_x[X]$  by the coordinates

$$\epsilon_x[X = \tilde{x}] = \begin{cases} 1 & \text{if } x = \tilde{x} \\ 0 & \text{else.} \end{cases} \quad (3)$$

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

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

$$\epsilon_x[X] = [0 \ \cdots \ 0 \ 1 \ 0 \ \cdots \ 0]^T, \quad (4)$$

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

Might need to discuss image enumerating maps!

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

**Definition 6 (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

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

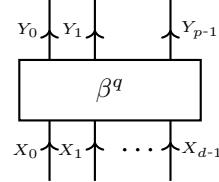
between these state sets. Then the basis encoding of  $q$  is a tensor

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

defined by

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

Basis encodings are normalized tensors and are thus depicted as decorations of directed edges in hypergraphs:



The entries of the basis encoding are defined by

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

Based on these concepts the main architecture can be defined.

**Definition 7 (Computation-Activation Network (CompActNets))** Given a function  $t : \times_{k \in [d]} [m_k] \rightarrow \mathbb{R}^p$ , and a hypergraph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with nodes  $[p] \subset \mathcal{V}$  containing the image coordinates of  $t$ , we define the by  $t$  computable and by  $\mathcal{G}$  activated family of distributions by

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

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

## 2.6 Continuous Variables

We can also allow for variables taking values in infinite sets such as  $\mathbb{R}$ . Contractions are then performed by generic integrations with respect to e.g. the Lebesgue measure.

## 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 (1971)) 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

As defined next, distributions  $\mathbb{P}$  over a discrete state space can be represented by tensors, where each entry corresponds to the probability of a corresponding state.

**Definition 8 (Joint Probability Distribution)** Let there be for each  $k \in [d]$  a categorical variable  $X_k$  taking values in  $[m_k]$ . A joint probability distribution of these categorical variables is a function

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

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

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

and which is normalized, that is

$$\langle \mathbb{P}[X_{[d]}] \rangle_{[\emptyset]} = 1.$$

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

**Example 5 (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  $\text{val}(Z) = [0, 1]$  and denote by  $X_{[d]}$   $d$  boolean variables. Then the family of coin toss distributions is modeled by the tensor  $\mathbb{P}[X_{[d]}, Z]$  with coordinates  $x_{[d]} \in \times_{k \in [d]} [2]$  and  $z \in [0, 1]$  by

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

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

$$\mathbb{P}[X_{[2]} | Z = z] = \begin{array}{c} x_1 \\ \vdots \\ 0 \\ \dashrightarrow \\ x_0 \downarrow_1^0 \\ \left[ \begin{array}{cc} (1-z)^2 & z \cdot (1-z) \\ z \cdot (1-z) & z^2 \end{array} \right] \end{array}.$$

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

**Definition 9** For any distribution  $\mathbb{P}[X, Y]$  the marginal distribution is given by the contraction

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

which is depicted by the diagram

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

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

$$\mathbb{P}[X|Y = y] := \begin{cases} \frac{1}{m} \cdot \mathbb{I}[X] & \text{if } \langle \mathbb{P}[X, Y = y] \rangle_{[\emptyset]} = 0 \\ \frac{1}{\langle \mathbb{P}[X, Y = y] \rangle_{[\emptyset]}} \cdot \mathbb{P}[X, Y = y] & \text{else} \end{cases}$$

and in the second case depicted by the diagram

$$\boxed{\mathbb{P}[X_0 | X_1 = x_1]} \quad \begin{matrix} x_0 \downarrow \\ x_0 \downarrow \\ \bullet \end{matrix} \quad := \quad \frac{\boxed{\mathbb{P}[X_0, X_1]} \quad \begin{matrix} x_0 \downarrow \\ x_1 \downarrow \\ \epsilon_{x_1} \end{matrix}}{\boxed{\mathbb{P}[X_0, X_1]} \quad \begin{matrix} x_0 \downarrow \\ x_1 \downarrow \\ \epsilon_{x_1} \end{matrix}} \quad =: \quad \boxed{\mathbb{P}[X_0 | X_1]} \quad \begin{matrix} x_0 \downarrow \\ x_1 \downarrow \\ \epsilon_{x_1} \end{matrix}.$$

### 3.2 The Independence mechanism: Graphical Model Factorization

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

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

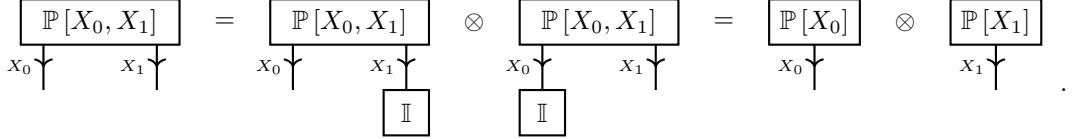
and therefore scales exponentially in the number of coordinates. To find efficient representation schemes of probability distributions by tensor networks, we need to exploit additional properties of the distribution. Independence leads to severe sparsifications of conditional probabilities and is therefore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

**Definition 10 (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 11 (Conditional Independence)** *Given a joint distribution of variables  $X_0$ ,  $X_1$  and  $X_2$ , such that  $\mathbb{P}[X_2]$  is positive. We say that  $X_0$  is independent of  $X_1$  conditioned on  $X_2$  if for any states  $x_0 \in [m_0]$ ,  $x_1 \in [m_1]$  and  $x_2 \in [m_2]$*

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

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

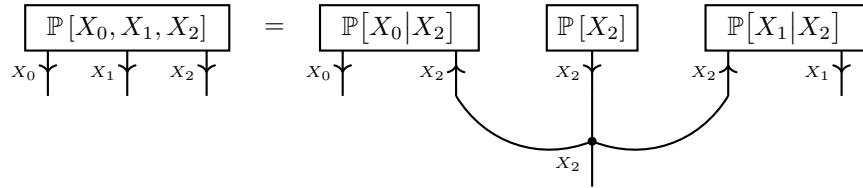
Conditional independence stated in Def. 11 has a close connection with independence stated in Def. 10. To be more precise,  $X_0$  is independent of  $X_1$  conditioned on  $X_2$ , if and only if  $X_0$  is independent of  $X_1$  with respect to any slice  $\mathbb{P}[X_0, X_1 | X_2 = x_2]$  of the conditional distribution  $\mathbb{P}[X_0, X_1 | X_2]$ .

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

**Corollary 12** *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*

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

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 13 (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*

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

We call the denominator

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

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

We 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 (2009)). The probabilistic graphical models are along that alternative dual to tensor networks Robeva and Seigal (2019); Glasser et al. (2019).

We can interpret the factors  $\tau[X_{[d]}]$  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 so far have defined Markov Networks as decomposed probability distributions, we now want to derive assumptions on a distribution assuring that such decompositions exist. As we will see, the sets of conditional independencies encoded by a hypergraph are captured by its separation properties, as we define next.

**Definition 14 (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 (2009).

**Definition 15 (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  $a, b \in e$ .

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

**Theorem 16 (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 6 (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

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

The corresponding hypergraph is the elementary graph, with respect to which any two disjoint subsets of nodes are separated (see Figure 4).

**Example 7** We consider a classical example of a graphical model (see (Koller and Friedman, 2009, Example 4.3)): A student of intelligence ( $X_I$ ) and SAT score ( $X_S$ ), is assigned a test of difficulty ( $X_D$ ), for which he gets a grade ( $X_G$ ) depending on which he gets a recommendation letter ( $X_L$ ) by its teacher. We make the following modelling assumptions:

- "The SAT score depends only on the students intelligence":  $(X_S \perp X_{\{D,G,L\}}) \mid X_I$
- "The recommendation letter depends only on the grade":  $(X_L \perp X_{\{D,I,S\}}) \mid X_G$

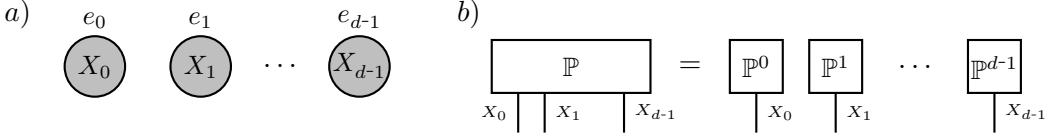


Figure 4: Decomposition of a probability distribution with independent variables (see Example 6). The independences are captured by the elementary hypergraph a), which edges contain single nodes. The corresponding tensor  $\mathbb{P}[X_d]$  is then represented by a Markov Network on the elementary hypergraph, where each factor is the marginal distribution of the corresponding variable.

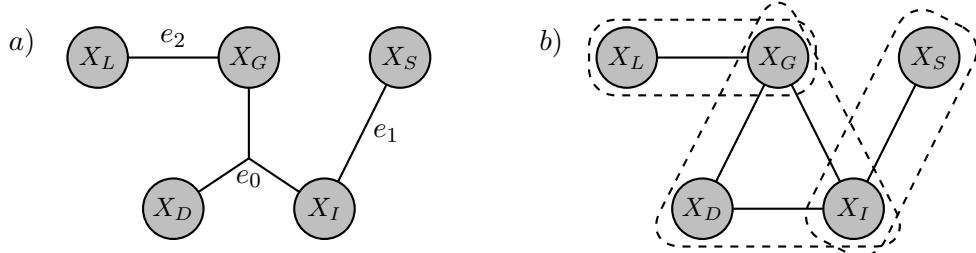


Figure 5: Hypergraph a) capturing the conditional independencies of the student example. The cliques of the node adjacency graph are highlighted in b), and coincide with hyperedges of the hypergraph. The hypergraph is therefore clique-capturing (see Def. 15).

The associated hypergraph capturing these conditional independencies is drawn in Figure 5 a).

Equivalently, Thm. 16 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

$$\mathbb{P}[X_V] = \langle \{\tau^e[X_e] : e \in \mathcal{E}\} \rangle_{[X_V|\emptyset]} .$$

Thus, the conditional-independence structure of  $\mathbb{P}[X_V]$  determines a global tensor-network decomposition of  $\mathbb{P}[X_V]$ . We refer to this correspondence between independence structure and tensor-network factorization as the *independence mechanism*.

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

We now present the computation mechanism of finding tensor-network decompositions of probability distributions.

**Definition 17** Let  $\mathbb{P}[X, Z]$  be a joint distribution of the  $m$ -dimensional variable  $X$  and the  $n$ -dimensional variable  $Z$  and let

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

be a statistic. We are interested in the distribution  $\mathbb{P}[X, Z, Y_t] = \langle \mathbb{P}[X, Z], \beta^t[Y_t, X] \rangle_{[X, Z, Y_t]}$ . 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. 17 is true if and only if

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

**Example 8 (Sufficient Statistics for the Probability)** Let  $Z$  be the value  $\mathbb{P}[X_{[d]} = x_{[d]}]$ , when drawing  $X_{[d]}$  from  $\mathbb{P}[X_{[d]}]$ . Then  $t$  is a sufficient statistic for  $Z = \mathbb{P}[X_{[d]}]$ , if for all  $y$  in the image of  $t$  we have

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

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

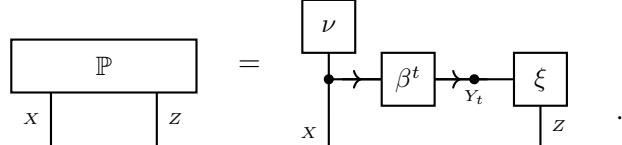
$$\mathbb{P}[X_{[d]}] = \langle \beta^t[Y_p, X_{[d]}], \xi[Y_p] \rangle_{[X_{[d]}]}.$$

Example 8 hints at a connection between sufficient statistics and decompositions into CompActNets. More generally, such decompositions are provided by the Fisher-Neyman Factorization Theorem.

**Theorem 18 (Fisher-Neyman Factorization Theorem)** Let  $\mathbb{P}$  be a joint distribution of variables  $X, Z$  with values  $\text{val}(X), \text{val}(Z)$ . Let there further be a finite set  $\text{val}(Y_t)$ . Then  $t : \text{val}(X) \rightarrow \text{val}(Y_t)$  is a sufficient statistic for  $Z$  if and only if there are tensors  $\nu[X]$  and  $\xi[Y_t, Z]$  such that

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

We depict this equation diagrammatically by



**Proof** Shown in more generality in the appendix, see Thm. 34. ■

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. 18 to embed such families in CompActNets.

**Corollary 19** Let  $\mathbb{P}[X_{[d]}|Z]$  be an arbitrary family of distributions of  $X_{[d]}$ , and  $t$  a sufficient statistic for  $Z$ . Then there is a tensor  $\nu[X_{[d]}]$  and a activation tensors  $\xi[Y_{[p]}, Z]$  such that for any  $z \in \text{val}(Z)$

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

**Example 9 (Order Statistic for Boolean Variables: Coin toss interpretation)** Let there be  $d$  boolean variables  $X_{[d]}$  and a family  $\{\mathbb{P}^\theta[X_{[d]}] : \theta \in \Theta\}$  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

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

When the order statistic is sufficient, the detailed order of the outcomes is 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. 18  $t^+$  is a sufficient statistic if and only if there are tensors  $\nu[X_{[d]}]$  and  $\xi[Y_+, \Theta]$  such that for each  $\theta \in \Theta$

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

The family of distributions, such that the variables  $X_{[d]}$  are i.i.d. with respect to each other (see Example 6) are the special case, where  $\nu[X_{[d]}] = \mathbb{I}[X_{[d]}]$  and the family is labeled by  $\theta \in [0, 1]$  such that for  $\theta \in (0, 1)$  and  $k \in [d + 1]$

$$\xi^\theta[Y_+ = k] = (1 - \theta)^{d-k} \cdot \theta^k ,$$

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

$$\xi^\theta[Y_+] = \begin{cases} \epsilon_0[Y_+] & \text{if } \theta = 0 \\ \epsilon_d[Y_+] & \text{if } \theta = 1 \end{cases} .$$

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

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

- The decomposition of  $\xi[Y_{[p]}]$  is called *activation network*.
- The decomposition of  $\beta^t[Y_{[p]}, X_{[d]}]$  is called *computation network*.

**Example 10 (Graphical Models as a special case of CompActNets)** For graphical models we take the identity statistic

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

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

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

and therefore, for any activation tensor  $\xi [Y_{[p]}]$  we obtain

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

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 Pitman (1936)) 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 (2022).

**Definition 20 (Exponential Family)** Given a base measure  $\nu$  and a statistic  $t : \times_{k \in [d]} [m_k] \rightarrow \mathbb{R}^p$  we enumerate for each coordinate  $l \in [p]$  the image  $\text{im}(t_l)$  by an interpretation map

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

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

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

and define the distribution

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

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

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

the exponential family to the statistic  $t$  and the base measure  $\nu$ .

Note that by construction each member of an exponential family is an element in a CompActNet with elementary activation cores, that is

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

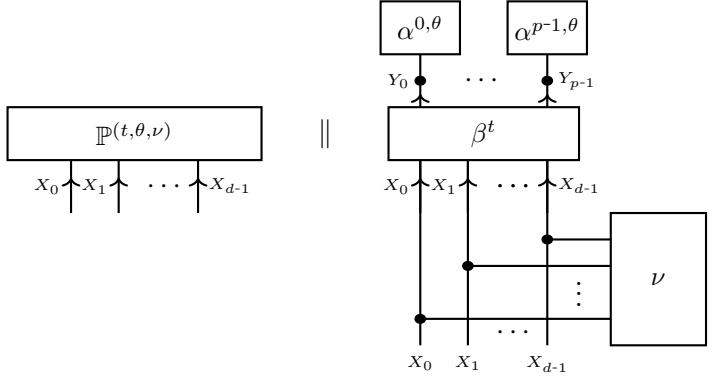


Figure 6: Tensor Network diagram of a member of an exponential family  $\mathbb{P}^{t,\nu} [X_{[d]} | \Theta = \theta]$  before normalization as an CompActNet with elementary activation, that is an element in  $\Lambda^{t,\text{EL},\nu}$ .

**Example 11 (Joint distributions of two booleans)** *In general, joint distribution of two Boolean variables  $X_0, X_1$  are  $2 \times 2$  matrices of non-negative coordinates summing to 1:*

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

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

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

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

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

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

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

*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*

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

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

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

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

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

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

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

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

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

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

$$\mathbb{P}[X_0, X_1] = \langle \beta^+ [Y_+, X_0, X_1], \xi [Y_+] \rangle_{[X_0, X_1] \mid \emptyset},$$

such that

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

where the normalization constant  $Z$  cancels out any multiplicative constant  $\lambda \in \mathbb{R} \setminus \{0\}$  in  $\xi$  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  $p_{1,1} = (\exp[\theta] (1 + \exp[\theta])^{-1})^2$ .

### 3.5 Efficient Contractions by Message Passing

We now formulate the Belief Propagation algorithm based on tensor network. We denote  $\mathcal{E}^\rightarrow$  to be all tuples  $(e_0, e_1)$  of hyperedges  $e_0, e_1 \in \mathcal{E}$  such that  $e_0 \neq e_1$  and  $e_0 \cap e_1 \neq \emptyset$ .

---

**Algorithm 1** Belief Propagation

---

**Require:** Tensor network  $\tau^G$  on a hypergraph  $G$

**Ensure:** Scheduler  $S$

---

Initialize  $S$  (differs in implementation)

Initialize messages

**while**  $S$  not empty **do**

    Take a  $(e_0, e_1)$  pair from  $S$

    Update the message

$$\chi_{e_0 \rightarrow e_1} [X_{e_0 \cap e_1}] = \langle \{\tau^{e_0} [X_{e_0}] \} \cup \{\chi_{e_2 \rightarrow e_0} [X_{e_2 \cap e_0}] : (e_2, e_0) \in \mathcal{E}^\rightarrow, e_2 \neq e_1\} \rangle_{[X_{e_0 \cap e_1}]}$$

    Update  $S$  (differs in implementation)

**end while**

**return** Messages  $\{\chi_{e_0 \rightarrow e_1} [X_{e_0 \cap e_1}] : (e_2, e_0) \in \mathcal{E}^\rightarrow\}$

---

**Example 12 (Continuation of Example 7)** Let us now exemplify the Belief Propagation Algorithm 1 on the Markov Network in the student example (see Example 7). The directions of the messages result from the hyperedge overlaps (see Figure 7 a) and the resulting directions  $\mathcal{E}^\rightarrow$  are sketched in Figure 7 b). The messages to  $\{(e_2, e_0), (e_0, e_2)\}$  are vectors of  $X_G$  and the messages  $\{(e_0, e_1), (e_1, e_0)\}$  are vectors of  $X_I$ .

Since the hyperedges are minimally connected, we can implement Algorithm 1 by a tree scheduler  $S$ :

- The scheduler is initialized with messages from leafs, in our example  $\{(e_2, e_0), (e_1, e_0)\}$
- Each message is placed exactly once on  $S$ , when at a hyperedge all but the reverse message have been received. In our example, after execution of  $(e_2, e_0)$  the message  $(e_0, e_1)$  is placed on  $S$  and after execution of  $(e_1, e_0)$  the message  $(e_0, e_2)$ .

In this implementation Algorithm 1 terminates after  $|\mathcal{E}^\rightarrow| = 4$  iterations of the While loop. The exact marginals of the edge variables are then

$$\begin{aligned} \mathbb{P}[X_L, X_G] &= \langle \tau^{e_2} [X_L, X_G], \chi_{e_0 \rightarrow e_2} [X_G] \rangle_{[X_L, X_G | \emptyset]} \\ \mathbb{P}[X_G, X_D, X_I] &= \langle \tau^{e_0} [X_G, X_D, X_I], \chi_{e_2 \rightarrow e_0} [X_G], \chi_{e_1 \rightarrow e_0} [X_I] \rangle_{[X_G, X_D, X_I | \emptyset]} \\ \mathbb{P}[X_I, X_S] &= \langle \tau^{e_1} [X_I, X_S], \chi_{e_0 \rightarrow e_1} [X_I] \rangle_{[X_I, X_S | \emptyset]} . \end{aligned}$$

## 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.

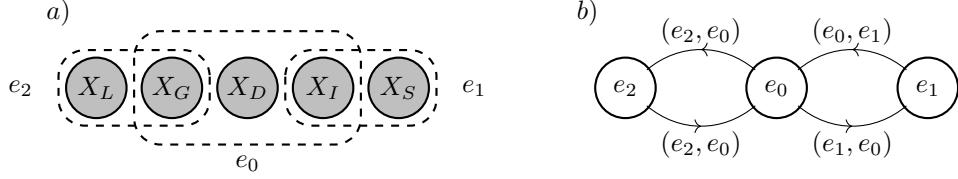


Figure 7: a) Sketch of the overlap of the edges, resulting in the message directions b)  $\mathcal{E}^{\rightarrow} = \{(e_2, e_0), (e_0, e_2), (e_0, e_1), (e_1, e_0)\}$ .

#### 4.1 Propositional Semantics by Boolean Tensors

**Definition 21** A propositional formula  $f [X_{[d]}]$  depending on  $d$  boolean variables  $X_k$  is a boolean-valued tensor

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

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

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

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

$$f [X_{[3]}] = \begin{array}{|c|c|c|} \hline & f & \\ \hline X_0 & | & | \\ & X_1 & X_2 \\ \hline \end{array} = \begin{array}{cc} \xrightarrow[X_1]{0 \quad 1} & \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \\ \downarrow & \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \\ \xrightarrow[X_2]{0 \quad 1} & \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \end{array}.$$

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.  $f [X_{[3]} = x_{[3]}] = 1$  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  $f [X_{[d]}]$  is equal to one at index  $x_{[d]}$  if and only if  $x_{[d]}$  is a model of  $f$ , i.e. fulfills the formula, a propositional formula can be written as the sum over the one-hot encodings of its models.

$$\begin{array}{c} \boxed{f} \\ \downarrow X_0 \quad \downarrow X_1 \quad \dots \quad \downarrow X_{d-1} \end{array} = \sum_{\substack{x_0, \dots, x_{d-1} \in \times_{k \in [d]} [2] \\ f(x_0, \dots, x_{d-1}) = 1}} \epsilon_{x_0} \quad \dots \quad \epsilon_{x_{d-1}}$$

This decomposition corresponds to the CP decomposition of a tensor.

**Example 14** For the formula described in Example 13, we have

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

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

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

**Model counts by contraction** Each coordinate 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  $\langle f \rangle_{[\emptyset]}$  counts the number of models of the propositional formula  $f$ . One can therefore decide the satisfiability of a formula by checking if  $\langle f \rangle_{[\emptyset]} > 0$ .

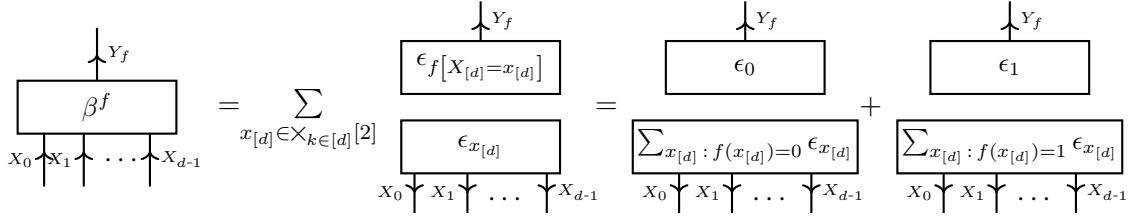
**Basis encoding** Representing booleans by elements in  $\{0, 1\}$  leads to the problem, that negation is an affine transformation and can not be represented by multilinear tensors. Therefore, instead of using this *coordinate calculus* an approach based on *basis calculus* is employed, which is explained 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. 5. Propositional formulas  $f$  can be expressed in terms of a tensor describing the mapping and its negation by

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

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

$$\beta^f [Y_f, X_{[d]}] = \epsilon_1 [Y_f] \otimes f [X_{[d]}] + \epsilon_0 [Y_f] \otimes \neg f [X_{[d]}]. \quad (6)$$

In our graphical notation this property is visualized by



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

**Example 15 (Logical Negation and Conjunction)** *The basis encodings of the negation  $\neg : [2] \rightarrow [2]$  is the matrix*

$$\beta^\neg [Y_\neg, X] = \begin{array}{c} Y_\neg \\ \overbrace{\phantom{X}}^{0 \rightarrow 1} \\ \downarrow \\ x_0 \downarrow^0 \\ \downarrow \\ x_1 \end{array} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

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

$$\beta^\wedge [Y_\wedge, X_0, X_1] = \begin{array}{c} X_1 \\ \overbrace{\phantom{X}}^{0 \rightarrow 1} \\ \downarrow \\ Y_\wedge \downarrow^0 \\ \downarrow \\ Y_\wedge \downarrow^1 \end{array} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \otimes \begin{array}{c} X_1 \\ \overbrace{\phantom{X}}^{0 \rightarrow 1} \\ \downarrow \\ X_0 \downarrow^0 \\ \downarrow \\ X_0 \downarrow^1 \end{array} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} + \begin{array}{c} X_1 \\ \overbrace{\phantom{X}}^{0 \rightarrow 1} \\ \downarrow \\ Y_\wedge \downarrow^0 \\ \downarrow \\ Y_\wedge \downarrow^1 \end{array} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{array}{c} X_1 \\ \overbrace{\phantom{X}}^{0 \rightarrow 1} \\ \downarrow \\ X_0 \downarrow^0 \\ \downarrow \\ X_0 \downarrow^1 \end{array} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \begin{array}{c} X_1 \\ \overbrace{\phantom{X}}^{0 \rightarrow 1} \\ \downarrow \\ X_0 \downarrow^0 \\ \downarrow \\ X_0 \downarrow^1 \end{array} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

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

$$\beta^\vee [Y_\vee, X_0, X_1] = \begin{array}{c} X_1 \\ \overbrace{\phantom{X}}^{0 \rightarrow 1} \\ \downarrow \\ X_0 \downarrow^0 \\ \downarrow \\ X_0 \downarrow^1 \end{array} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

**Interpretation as CompActNets** The propositional formula and its negation can be represented by that tensor by

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

Both  $f$  and  $\neg f$  are thus Computation-Activation Networks to the statistic  $\{f\}$  and the hard activation tensor  $\epsilon_1 [Y_f]$ , respectively  $\epsilon_0 [Y_f]$ . 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 22** *Let  $f [X_{[d]}]$  be a composition of a  $p$ -ary connective formula  $\circ$  and propositional formulas  $f_l [X_{[d]}]$ , where  $l \in [p]$ , i.e. for  $x_{[d]} \in \times_{k \in [d]} [2]$  we have*

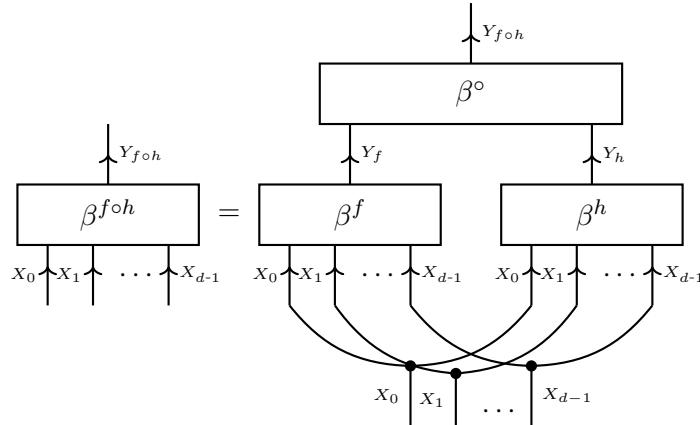
$$f [X_{[d]} = x_{[d]}] = \circ (f_0 [X_{[d]} = x_{[d]}], \dots, f_{p-1} [X_{[d]} = x_{[d]}]).$$

Then we have

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

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

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



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

**Definition 23** *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.  $|e^{\text{out}}| = 1$
- 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  $e^{\text{in}} \neq \emptyset$  is decorated by a propositional formula

$$\circ_v [Y_{e^{\text{in}}} : \bigtimes_{v \in e^{\text{in}}} [2] \rightarrow [2]]$$

- 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

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

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

**Theorem 24** For any syntactical hypergraph  $\mathcal{G}$  with composition  $f$  we have

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

**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. 22.  $\blacksquare$

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

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

If  $\mathcal{KB} \models \neg f$  holds (i.e.  $\langle \mathcal{KB}, f \rangle_{[\emptyset]} = 0$ ), we say that  $\mathcal{KB}$  contradicts  $f$ .

Classically (see e.g. Russell and Norvig (2021)) entailment in propositional logics is defined as the unsatisfiability of  $\mathcal{KB} \wedge \neg f$ . This is equivalent to Def. 25, since  $\langle \mathcal{KB}, \neg f \rangle_{[\emptyset]} = 0$  is equivalent to  $\langle \mathcal{KB} \wedge (\neg f) \rangle_{[\emptyset]} = 0$ , 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 16 ( $n^2 \times n^2$  Sudoku)** We index the rows and the columns by tuples  $(r0, r1)$  and  $(c0, c1)$ , 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  $r_0, r_1, c_0, c_1 \in [n]$  and  $i \in [n^2]$  we then define an atomic variable  $X_{r_0, r_1, c_0, c_1, i} \in \{0, 1\}$  indicating whether in the row  $(r_0, r_1)$  and column  $(c_0, c_1)$  the number  $i$  is written. The Sudoku rules then amount to the formula

$$\mathcal{KB}^n := \left( \bigwedge_{r_0, r_1, c_0, c_1 \in [n]} \left( \bigoplus_{i \in [n^2]}^{(1)} X_{r_0, r_1, c_0, c_1, i} \right) \right) \wedge \left( \bigwedge_{r_0, r_1 \in [n], i \in [n^2]} \left( \bigoplus_{c_0, c_1 \in [n]}^{(1)} X_{r_0, r_1, c_0, c_1, i} \right) \right) \wedge \\ \left( \bigwedge_{c_0, c_1 \in [n], i \in [n^2]} \left( \bigoplus_{r_0, r_1 \in [n]}^{(1)} X_{r_0, r_1, c_0, c_1, i} \right) \right) \wedge \left( \bigwedge_{r_0, c_0 \in [n], i \in [n^2]} \left( \bigoplus_{r_1, c_1 \in [n]}^{(1)} X_{r_0, r_1, c_0, c_1, i} \right) \right),$$

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  $E^{\text{start}}$  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  $(r_0, r_1, c_0, c_1, i) \notin E^{\text{start}}$  whether the Sudoku rules and the initial board imply that the atomic variable  $X_{r_0, r_1, c_0, c_1, i}$  (i.e. assignment to the board) is true

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

or false

$$\mathcal{KB} \wedge \left( \bigwedge_{(r_0, r_1, c_0, c_1, i) \in E^{\text{start}}} X_{r_0, r_1, c_0, c_1, i} \right) \models \neg X_{r_0, r_1, c_0, c_1, i}. \quad (7)$$

If and only if the Sudoku has a unique solution given the initial board assignment  $E^{\text{start}}$ , exactly one of these entailment statements holds for each  $(r_0, r_1, c_0, c_1, i) \notin E^{\text{start}}$ . Deciding which is equivalent to solving of the Sudoku.

For a more detailed example, let  $n = 2$  and

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

We visualize this evidence by writing  $i + 1$  in a grid cell  $(r_0, r_1, c_0, c_1)$  to indicate that  $(r_0, r_1, c_0, c_1, i) \in E^{\text{start}}$ :

1		3	2
	2		
		4	
4	3		

We will later demonstrate in Example 16 a solution algorithm to solve this instance, after we have derived sparse tensor network representations in Example 18.

#### 4.4 Efficient Representation of Knowledge Bases

We now investigate the representation of propositional knowledge bases  $\mathcal{KB} = \{f_l : l \in [p]\}$ , which are sets of propositional formulas  $f_l$ . The conjunction of these formulas is the knowledge base formula

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

To show efficient representations we will use the following identities.

**Lemma 26 (Computation Network Symmetries)** *We have for the  $d$ -ary  $\wedge$ -connective (where  $d \in \mathbb{N}$ ) and the unary  $\neg$ -connective that*

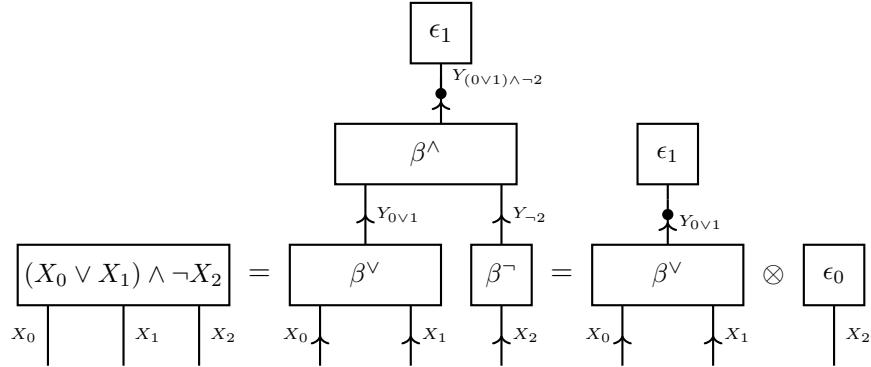
$$\langle \epsilon_1[Y], \beta^\wedge[Y, X_{[d]}] \rangle_{[X_{[d]}]} = \bigotimes_{k \in [d]} \epsilon_1[X_k] \quad \text{and} \quad \langle \epsilon_1[Y], \beta^\neg[Y, X] \rangle_{[X]} = \epsilon_0[X].$$

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

**Example 17 (Computation Network Symmetries (Continuation of Example 13))**  
For the propositional formula

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

we can write the formula in terms of a Computation-Activation Network with activation tensor  $\epsilon_1$  and computation network decomposed by the basis encodings. First, it is written with one activation vector. Second, we see that it can also be interpreted with multiple features.



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

**Theorem 27** *For any knowledge base  $\mathcal{KB}[X_{[d]}] = \bigwedge_{l \in [p]} f_l[X_{[d]}]$  it holds that*

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

**Proof** With Lem. 26 we have

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

■

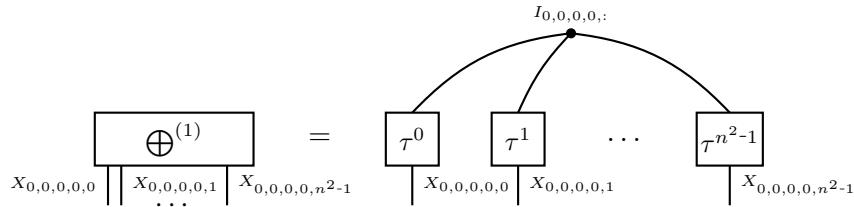
**Example 18 (Sparse representation of Sudoku rule Knowledge Base)** We now exploit Thm. 27 to find efficient tensor network representation of the Sudoku knowledge base from Example 16. We directly get, that the knowledge base  $\mathcal{KB}^n$  of Sudoku rules is a tensor network of the  $4 \cdot n^4$  constraint formulas using the  $n^2$ -ary connective  $\oplus^{(1)}$ , and the evidence  $E^{\text{start}}$  can be encoded by vectors  $\epsilon_1 [X_{(r_0, r_1, c_0, c_1, i)}]$ . To get a representation by vectors matrices instead of tensors of order  $n^2$ , we introduce a hidden variable  $I$  taking values in  $[n^2]$  for each of the constraints, one can further increase the sparsity of the representation. With the usage of matrices

$$\tau^k [X_k, I] = \epsilon_0 [X_k] \otimes \mathbb{I}[I] + (\epsilon_1 [X_k] - \epsilon_0 [X_k]) \otimes \epsilon_k [I]$$

we have the decomposition

$$\bigoplus^{(1)} [X_{[n^2]}] = \left\langle \{\tau^k [X_k, I] : k \in [n^2]\} \right\rangle_{[X_{[n^2]}]},$$

which is a CP decomposition (for the example of the position constraint at  $(r_0, r_1, c_0, c_1) = (0, 0, 0, 0)$ ) depicted as:



We model the Sudoku Knowledge Base as a tensor network on a hypergraph  $\mathcal{G}^{\text{Sudoku}, n}$

- $n^6 + 4 \cdot n^4$  Nodes by  $n^6$  categorical variables  $X_{(r_0, r_1, c_0, c_1, i)}$  and by  $4 \cot n^4$  decomposition variables to the constraints
- $4 \cdot n^6$  Edges

$$\begin{aligned}
\mathcal{E} = \bigcup_{r_0, r_1, c_0, c_1 \in [n]} &\{ \{X_{(r_0, r_1, c_0, c_1, i)}\}, \{X_{(r_0, r_1, c_0, c_1, i)}, I_{r_0, r_1, c_0, c_1, :}\}, \{X_{(r_0, r_1, c_0, c_1, i)}, I_{r_0, r_1, :, :, i}\}, \\
&\{X_{(r_0, r_1, c_0, c_1, i)}, I_{:, :, c_0, c_1, i}\}, \{X_{(r_0, r_1, c_0, c_1, i)}, I_{r_0, :, c_0, :, i}\}
\end{aligned}$$

We denote the decomposition variables to the position, row, column and square constraints by  $I_{r0,r1,c0,c1,:}$ ,  $I_{r0,r1,:,:,:i}$ ,  $I_{:,:,c0,c1,i}$  and  $I_{r0,:,:c0,:,:i}$ .

Each edge containing a decomposition variable is decorated by a matrix  $\tau^k X, I$  corresponding to an core in the CP decomposition of a constraint. Here  $k$  is determined by the tuple  $(r0, r1, c0, c1, i)$  and the type of the constraint (for example, for the variable  $X_{(0,1,1,2,1)}$  and the row constraint  $I_{(0,1,:,:,:1)}$  we have  $k = 1 \cdot n + 2$ . We further assign to each edge containing a single variable  $\{X_{(r0,r1,c0,c1,i)}\}$  a either the vector  $\epsilon_1 [X_{(r0,r1,c0,c1,i)}]$  if  $(r0, r1, c0, c1, i) \in E^{\text{start}}$  or the trivial vector  $\mathbb{I} [X_{(r0,r1,c0,c1,i)}]$ .

#### 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 28 (Monotonicity 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  $\langle \tilde{\mathcal{KB}}[X_{[d]}], \neg f[X_{[d]}] \rangle_{[\emptyset]} = 0$  and thus  $\langle \tilde{\mathcal{KB}}[X_{[d]}], \neg f[X_{[d]}] \rangle_{[X_{[d]}]} = 0 [X_{[d]}]$ . Denoting by  $\mathcal{KB}/\tilde{\mathcal{KB}}$  the conjunctions of formulas in  $\mathcal{KB}$  not in  $\tilde{\mathcal{KB}}$ , we have

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

■

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 furthermore add entailed formulas to the knowledge base without the latter, as we show next.

**Theorem 29 (Invariance of adding Entailed Formulas)** *If  $\mathcal{KB} \models f$  then*

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

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

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

■

One can understand entailment 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. Thm. 29 justifies this intuition in our tensor network formalism.

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. Let us now refine Algorithm 1 to this situation. Since we are only interested in the support of the contractions, we schedule new messages in the direction  $(e_0, e_1)$ , once the support of a message received at  $e_0$  has been changed. Note that such a scheduling system is guaranteed to converge, since there can only be a finite number of message changes. We further directly reduce the computation of messages to their support and call the resulting Algorithm 2 Constraint Propagation.

---

**Algorithm 2** Constraint Propagation

---

**Require:** Tensor network  $\tau^G$  on a hypergraph  $G$

---

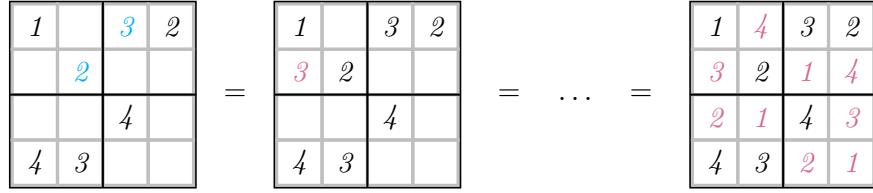
```

Initialize a queue  $S = \mathcal{E}^\rightarrow$  of message directions
Initialize messages  $\chi_{e_0 \rightarrow e_1}[X_{e_0 \cap e_1}] = \mathbb{I}[X_{e_0 \cap e_1}]$  for  $(e_0, e_1) \in \mathcal{E}^\rightarrow$ 
while  $S$  not empty do
    Pop a  $(e_0, e_1)$  pair from  $S$ 
    Compute
     $\chi_{e_0 \rightarrow e_1}[X_{e_0 \cap e_1}] = \mathbb{I}_{\neq 0} \left( \langle \{\tau^{e_0}[X_{e_0}]\} \cup \{\chi_{e_2 \rightarrow e_0}[X_{e_2 \cap e_0}] : (e_2, e_0) \in \mathcal{E}^\rightarrow, e_2 \neq e_1\} \rangle_{[X_{e_0 \cap e_1}]} \right)$ 
    if  $\tau[X_{e_0 \cap e_1}] \neq \chi_{e_0 \rightarrow e_1}[X_{e_0 \cap e_1}]$  then
        Update the message:  $\chi_{e_0 \rightarrow e_1}[X_{e_0 \cap e_1}] := \tau[X_{e_0 \cap e_1}]$ 
        Add  $S = S \cup \{(e_1, e_2) : (e_1, e_2) \in \mathcal{E}^\rightarrow\}$ 
    end if
end while
return Messages  $\{\chi_{e_0 \rightarrow e_1}[X_{e_0 \cap e_1}] : (e_2, e_0) \in \mathcal{E}^\rightarrow\}$ 

```

---

**Example 19 (Constraint Propagation for the Sudoku of Example 16)** We iteratively solve a Sudoku puzzle by determining a possible value based on neighboring cells, rows and squares (using Thm. 28) and adding to our knowledge (using Thm. 29). 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:



To illustrate the first reasoning step of assigning  $X_{0,1,0,0,2}$  we make the following entailment steps applying Thm. 28. We also depict in Figure 8 the corresponding messages in the Constraint Propagation Algorithm on the hypergraph  $\mathcal{G}^{\text{Sudoku},n}$ .

- 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  $(0, 1, 0, 1)$ . This entailment step is performed by 4 consecutive messages (see  $\chi^{0,[4]}$  in Figure 8) involving decomposition cores of the position constraint in the position  $(r_0, r_1, c_0, c_1) = (0, 0, 0, 0)$ .

- 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

$$\left( \bigoplus_{c_0, c_1 \in [n]}^{(1)} X_{0,0,c_0,c_1,2} \right) \wedge X_{0,0,1,0,2} \models \neg X_{0,0,0,0,2} \wedge \neg X_{0,0,0,1,2},$$

That is, that the number 3 is neither in the cell  $(0, 0, 0, 0)$  nor in  $(0, 0, 0, 1)$ . This entailment step is performed by 7 consecutive messages (see  $\chi^{1,[7]}$  in Figure 8) involving decomposition cores of the constraint to the number  $i = 3$  in the first row  $(r_0, r_1) = (0, 0)$ .

We add these formulas to our knowledge base (justified by Thm. 29) and use the rule, that 3 appears exactly once in the first square

$$\left( \bigoplus_{r_1, c_1 \in [n]}^{(1)} X_{0,r_1,0,c_1,2} \right) \wedge (\neg X_{0,1,0,1,2}) \wedge (\neg X_{0,0,0,0,2} \wedge \neg X_{0,0,0,1,2}) \models X_{0,1,0,0,2}.$$

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. This entailment step is performed by four consecutive messages (see  $\chi^{2,[4]}$  in Figure 8) along the directions  $(X_{0,1,0,1,2}, I_{0,:,0,:,2})$ ,  $(X_{0,0,0,0,2}, I_{0,:,0,:,2})$ ,  $(X_{0,0,0,1,2}, I_{0,:,0,:,2})$  and  $(I_{0,:,0,:,2}, X_{0,1,0,0,2})$ .

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

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

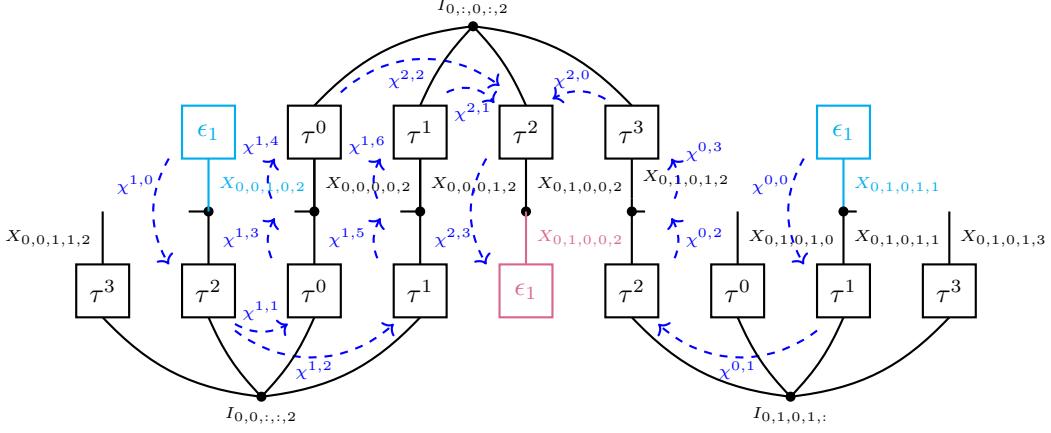


Figure 8: Tensor network decomposition of three rules for the  $2^2 \times 2^2$  Sudoku rules, namely to the number 3 appearing once in the  $(0, 0)$ -square (above), the number 3 appearing once in the  $(0, 0)$ -row (below left) and a unique number appearing at the  $(0, 1, 0, 1)$ -position (below right). The tensor network therefore showcasts 3 out of  $2 \cdot 2^4 = 128$  rules in the Sudoku knowledge base (see Example 18). The evidence of the number 3 assigned at the position  $(0, 0, 1, 0)$  is sketched on by a basis vector  $\epsilon_1$  on the left side, and the number 2 assigned at position  $(0, 1, 0, 1)$  analogously on the right side. During Constraint Propagation Algorithm 2 on the hypergraph of Sudoku rules and evidence (see Example 19), this evidence leads to non-trivial messages  $\chi$ , which propagate the constraints by partial entailment steps and imply that  $X_{0,1,0,0,2}$  is true, i.e. that at the position  $(0, 1, 0, 0)$  the number 3 needs to be assigned.

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

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

## 5 Hybrid Logic Networks

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

### 5.1 Parametrization

**Definition 30 (Hybrid Logic Network (HLN))** Given a boolean statistic  $t$  we call any element of  $\Lambda^{t,\text{EL}}$  a Hybrid Logic Network. The extended canonical parameter set to  $t$  is the set

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

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

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

where the activation core is

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

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<sup>1</sup>

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

further for all  $l \in A$

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

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

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

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

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

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

**Example 20 (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

---

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

- Exactly one account must be booked.
- 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

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

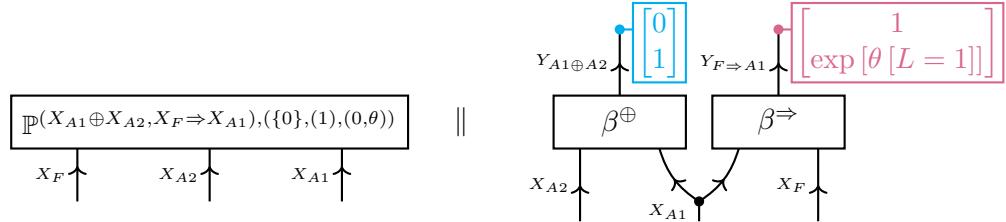
Since the first feature is hard, the "soft logic activation tensor" should be invariant under the first coordinate of the canonical parameter and we set  $\theta[L=0] = 0$ . We choose the soft parameters as  $\theta[L] = [0, \theta[L=1]]^\top$  to achieve

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

The activation tensor of the hybrid network then has the form

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

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



The resulting Hybrid Logic Network is a tensor  $\mathbb{P}^{t,(A,y_A,\theta)}[X_{A1}, X_{A2}, X_F]$  of order 3. With  $Y_{F⇒A1} = 1$  for  $F = 0$  and any  $A_1$  it has the coordinates

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

## 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  $((x_0^j, \dots, x_{d-1}^j) : j \in [m])$  consisting of  $m$  independent and identically distributed samples from an unknown distribution, we want to find a Hybrid Logic Network  $\mathbb{P}^{t,(A,y_A,\theta)}[X_{[d]}]$  that maximizes the data likelihood

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

We notice that this is  $\infty$  if and only if there is a data point  $j \in [m]$  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

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

by

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

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

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

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

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

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 in Algorithm 3. 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. (2008)).

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 3** 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  $\Lambda^{\mathcal{F},\text{EL}}$ 

Set

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

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

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

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 21 (Continuation of Example 20)** Let us recall the statistic of Example 20 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  $f_1 = X_F \Rightarrow X_{A1}$ 

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

Then Algorithm 3 yields with a reasonable convergence criterion choice (such as finite iterations or convergence of  $\theta [L]$ )

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

To derive this, we notice that Algorithm 3 treats formula  $f_0$  as a hard constraint and assigns  $A = \{0\}$  and  $y_A = 1$ . In the While loop we then have for the formula  $f_1$

$$\tau [Y_1] = \left\langle \epsilon_1 [Y_0], \beta^{f_0} [Y_0, X_F, X_{A1}, X_{A2}], \beta^{f_1} [Y_1, X_F, X_{A1}, X_{A2}] \right\rangle_{[Y_1]} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

since  $f_0$  has 4 models, of which 3 are also models of  $f_1$  and 1 is instead a model of  $\neg f_1$ . Notice, that the tensor  $\tau [Y_1]$  will not change in any further iteration of the While and the parameter  $\theta [L = 1]$  will therefore stay constant until the termination of the algorithm.

### 5.3 Entailment by Hybrid Logic Networks

Let us now demonstrate a further usage of our unified treatment of probabilistic and logical models, by investigating a generalized concept of entailment. Entailment can be generalized to probabilistic models, by deciding whether a propositional formula is always satisfied given a probabilistic model.

**Theorem 31** Let  $\mathbb{P}^{t,(A,y_A,\theta)} [X_{[d]}]$  be a Hybrid Logic Network and  $h [X_{[d]}]$  a propositional formula. Then we have the probabilistic entailment of  $h$  by  $\mathbb{P}^{t,(A,y_A,\theta)}$ , that is

$$\left\langle \mathbb{P}^{t,(A,y_A,\theta)} [X_{[d]}], h [X_{[d]}] \right\rangle_{[\emptyset]} = 1,$$

if and only if

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

where

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

**Proof** We have

$$\left\langle \mathbb{P}^{t,(A,y_A,\theta)} [X_{[d]}], h [X_{[d]}] \right\rangle_{[\emptyset]} = 1$$

if and only if

$$\left\langle \left( \mathbb{I} [X_{[d]}] - \mathbb{I}_{\neq 0} \left( \mathbb{P}^{t,(A,y_A,\theta)} [X_{[d]}] \right) \right), h [X_{[d]}] \right\rangle_{[\emptyset]} = 0$$

We notice, that  $f^{t,(A,y_A)} [X_{[d]}]$  is the indicator tensor for the support of  $\mathbb{P}^{t,(A,y_A,\theta)} [X_{[d]}]$ . It therfore holds that

$$\begin{aligned} & \left\langle \left( \mathbb{I} [X_{[d]}] - \mathbb{I}_{\neq 0} \left( \mathbb{P}^{t,(A,y_A,\theta)} [X_{[d]}] \right) \right), h [X_{[d]}] \right\rangle_{[\emptyset]} \\ &= \left\langle h [X_{[d]}] \right\rangle_{[\emptyset]} - \left\langle f^{t,(A,y_A)} [X_{[d]}], h [X_{[d]}] \right\rangle_{[\emptyset]} \\ &= \left\langle h [X_{[d]}] \right\rangle_{[\emptyset]} - \left\langle f^{t,(A,y_A)} [X_{[d]}], (\mathbb{I} [X_{[d]}] - \neg h [X_{[d]}]) \right\rangle_{[\emptyset]} \\ &= \left\langle f^{t,(A,y_A)} [X_{[d]}], h [X_{[d]}] \right\rangle_{[\emptyset]}. \end{aligned}$$

By Def. 25 this vanishes if and only if  $f^{t,(A,y_A)} \models h$ . ■

**Example 22 (Continuation of Example 21)** Let us consider again the Hybrid Logic Network  $\mathbb{P}^{(X_{A1} \oplus X_{A2}, X_F \Rightarrow X_{A1}), (\{0\}, (1), (0, \ln[3]))}$  from Example 21 and assume we want to decide the probabilistic entailment of the formula

$$h[X_{A1}, X_{A2}, X_F] = \neg X_{A1} \vee \neg X_{A2} \vee \neg X_F,$$

which has all states but  $(1, 1, 1)$  as a model (and is therefore referred to as a maxterm). Using Thm. 31 we have that

$$\left\langle \mathbb{P}^{(X_{A1} \oplus X_{A2}, X_F \Rightarrow X_{A1}), (\{0\}, (1), (0, \ln[3]))} [X_{A1}, X_{A2}, X_F], h[X_{A1}, X_{A2}, X_F] \right\rangle_{[\emptyset]} = 1$$

if and only if  $X_{A1} \oplus X_{A2} \models \neg X_{A1} \vee \neg X_{A2} \vee \neg X_F$ . By Def. 25 this entailment holds, since by the De-Morgan rule

$$\begin{aligned} \langle X_{A1} \oplus X_{A2}, \neg(\neg X_{A1} \vee \neg X_{A2} \vee \neg X_F) \rangle_{[\emptyset]} &= \langle X_{A1} \oplus X_{A2}, X_{A1}, X_{A2}, X_F \rangle_{[\emptyset]} \\ &= \langle X_F \rangle_{[\emptyset]} \cdot \langle X_{A1} \oplus X_{A2}, X_{A1}, X_{A2} \rangle_{[\emptyset]} \\ &= 0. \end{aligned}$$

We thus conclude, that  $h$  is probabilistically entailed by  $\mathbb{P}^{(X_{A1} \oplus X_{A2}, X_F \Rightarrow X_{A1}), (\{0\}, (1), (0, \ln[3]))}$ .

## 6 Implementation

Needs a major overwork: Present tnreason architecture, capabilities, add examples from reasoning.

The architecture can be conveniently implemented with the python package tnreason. Multiple examples including graph-coloring, sat problems, Sudoku, and temporal clue are available<sup>2</sup>. To emphasize the intuitive implementation of CANs, a code snippet for the implementation of a CAN for the sat problem  $f[X_a = a, X_b = b, X_c = c] = (a \vee b) \wedge \neg c$  described in example 15 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")}
```

---

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

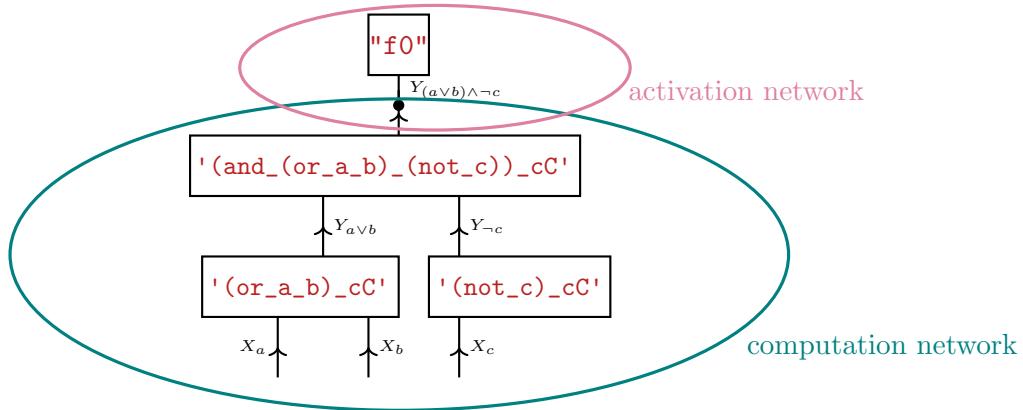
The generated `computationCores` is a dictionary with the following keys and tensors of given shapes as expected in example 15.

```
'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 soft 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 15, the CAN then has the following form. [Should we use the engine.draw\(\)](#) for that?



The other representation in example 15 can also be implemented. Both architectures can be found the linked notebook<sup>3</sup>. Note that more efficient representations of this network are possible and the one described here is mainly for pedagogic purposes. Furthermore the right implementation of the formula can be checked by contractions.

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

3. <https://colab.research.google.com/drive/14knFuMJHI683DAmUgJ-G10MQFueoXR6q#scrollTo=vr0YBViZhX2S>

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

This work has treated the representation of several models in tensor networks. Especially, probability distributions over discrete sets and propositional formulas can be represented in the introduced structure. Multiple properties, such as independence of variables for the distributions and connections of subsets for the propositional formulas, can be directly encoded into the architecture leading to sparse, memory-efficient representations. The exact representation also encourages the carrying over of analysis of the mathematical concepts to the CompActNets.

Model inference such as the calculating marginal distributions and deciding entailment are formulated by tensor network contractions. The computation of these contractions is however often a bottleneck. This bottleneck is related to the NP-hardness of probabilistic inferences in graphical models (see Koller and Friedman (2009)) and of logical reasoning (see Russell and Norvig (2021)). Approximation schemes to overcome this bottleneck can be summarized under the umbrella of variational inference (see Wainwright and Jordan (2008)), such as loopy message passing schemes and mean field methods. We in this work presented message-passing schemes as belief propagation in probability theory and syntactical inference algorithms in logics. We can understand them as approximation of (potentially intractible) contractions and will dedicate future work to study them in the tñreason formalism. While these schemes are developed either for graphical models (sum-product algorithms, belief-propagation) or more general exponential families (expectation-propagation), we anticipate to derive them in future work for generic CompActNets. Further frequently applied schemes are particle-based inference schemes such as Gibbs sampling.

Beyond the theoretical integration of differentiable architectures into the transformer architecture of Large Language Models Vaswani et al. (2017), the CompActNets framework offers an immediate practical application as a verifiable reasoning engine for AI agents in high-stakes domains such as regulatory compliance, clinical decision support or accounting. By leveraging the framework's inherent flexibility, Large Language Models can be adapted to function as semantic translators that dynamically construct problem-specific tensor networks in the form of CompActNets from natural language descriptions, effectively treating the reasoning engine as an external tool. This approach mitigates the hallucination risks of probabilistic models by delegating complex logical execution to the exact linear algebra of the tensor network, ensuring that the inference process is both rigorous and reproducible. Consequently, this synergy enables the deployment of reliable AI systems where the intuitive power of the LLM is grounded by the explainable, instance-adaptive topology of the Computation-Activation Network.

### **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. *Physical Review Letters*, 59(7):799–802, August 1987. doi: 10.1103/PhysRevLett.59.799. URL <https://link.aps.org/doi/10.1103/PhysRevLett.59.799>. Publisher: American Physical Society.
- Alejandro Mata Ali. Explicit Solution Equation for Every Combinatorial Problem via Tensor Networks: MeLoCoToN, February 2025. URL <http://arxiv.org/abs/2502.05981>.
- Artur S. Avila Garcez and Gerson Zaverucha. The Connectionist Inductive Learning and Logic Programming System. *Applied Intelligence*, 11(1):59–77, July 1999. ISSN 1573-7497. doi: 10.1023/A:1008328630915. URL <https://doi.org/10.1023/A:1008328630915>.
- Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss Markov random fields and probabilistic soft logic. *J. Mach. Learn. Res.*, 18(1):3846–3912, January 2017. ISSN 1532-4435.
- Samy Badreddine, Artur d’Avila Garcez, Luciano Serafini, and Michael Spranger. Logic Tensor Networks. *Artificial Intelligence*, 303:103649, February 2022. 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, June 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. *Physical Review Research*, 7(1):013063, January 2025. doi: 10.1103/PhysRevResearch.7.013063. URL <https://link.aps.org/doi/10.1103/PhysRevResearch.7.013063>. Publisher: American Physical Society.
- George Casella and Roger Berger. *Statistical Inference*. Cengage Learning, Pacific Grove, Calif, June 2001. ISBN 978-0-534-24312-8.
- P. Clifford and J. M. Hammersley. Markov fields on finite graphs and lattices. *Unpublished*, 1971. 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. Jeju, South Korea, 2024.
- Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, Hoboken, N.J, 2nd edition edition, September 2006. ISBN 978-0-471-24195-9.

Pedro Domingos. Tensor Logic: The Language of AI, October 2025. URL <http://arxiv.org/abs/2510.12269>. arXiv:2510.12269 [cs].

Martin Eigel, Max Pfeffer, and Reinhold Schneider. Adaptive stochastic Galerkin FEM with hierarchical tensor representations. *Numerische Mathematik*, 136(3):765–803, July 2017. ISSN 0945-3245. 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, Arlington, Virginia, USA, July 2008. 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, May 2019. URL <http://arxiv.org/abs/1905.06088>.

Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning*. MIT Press, September 2019. 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. *Advances in Neural Information Processing Systems*, 32, 2019.

Alex Goessmann. *Uniform Concentration of Tensor and Neural Networks: An Approach towards Recovery Guarantees*. PhD Thesis, Technische Universität Berlin, Berlin, 2021. URL <https://depositonce.tu-berlin.de/handle/11303/15990>.

Alex Goessmann. The Tensor-Network Approach to Efficient and Explainable AI, 2025. URL <https://github.com/EnexaProject/enexa-tensor-reasoning-documentation/>.

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, 2020.

Nikita Gourianov, Peyman Givi, Dieter Jaksch, and Stephen B. Pope. Tensor networks enable the calculation of turbulence probability distributions. *Science Advances*, 11(5):eads5990, January 2025. doi: 10.1126/sciadv.ad5990. URL <https://www.science.org/doi/10.1126/sciadv.ad5990>. Publisher: American Association for the Advancement of Science.

W. Hackbusch and S. Kühn. A New Scheme for the Tensor Representation. *Journal of Fourier Analysis and Applications*, 15(5):706–722, October 2009. ISSN 1531-5851.

Wolfgang Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*. Springer Series in Computational Mathematics. Springer-Verlag, Berlin Heidelberg, 2012. ISBN 978-3-642-28026-9. doi: 10.1007/978-3-642-28027-6.

- 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*, pages 374–386. Springer Nature Switzerland, Cham, 2025. ISBN 978-3-031-92366-1.
- Frank L. Hitchcock. The Expression of a Tensor or a Polyadic as a Sum of Products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927. ISSN 1467-9590. doi: <https://doi.org/10.1002/sapm192761164>.
- Sepp Hochreiter. Toward a broad AI. *Communications of the ACM*, 65(4):56–57, January 2022. 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, Cambridge, Mass., 1. edition edition, July 2009. ISBN 978-0-262-01319-2.
- J. Landsberg. *Tensors: Geometry and Applications*, volume 128 of *Graduate Studies in Mathematics*. American Mathematical Society, December 2011. 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. *Commun. ACM*, 61(10):36–43, September 2018. ISSN 0001-0782. doi: 10.1145/3233231. URL <https://dl.acm.org/doi/10.1145/3233231>.
- Giuseppe Marra, Sebastijan Dumančić, Robin Manhaeve, and Luc De Raedt. From statistical relational to neurosymbolic artificial intelligence: A survey. *Artificial Intelligence*, 328:104062, March 2024. 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 Stationery Office, London, 1959. URL <http://www-formal.stanford.edu/jmc/mcc59.html>.
- Kevin P. Murphy. *Probabilistic Machine Learning: An Introduction*. The MIT Press, Cambridge, Massachusetts London, England, March 2022. ISBN 978-0-262-04682-4.
- Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE*, 104(1): 11–33, January 2016. 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, September 2019. ISSN 2522-5820. doi: 10.1038/s42254-019-0086-7.
- I. V. Oseledets. Tensor-Train Decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, January 2011. ISSN 1064-8275. doi: 10.1137/090752286. URL <https://pubs.siam.org/doi/10.1137/090752286>. Publisher: Society for Industrial and Applied Mathematics.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, s.l., September 1988. ISBN 978-1-55860-479-7.

Roger Penrose. *Spinors and Space-Time: Volume 1, Two-Spinor Calculus and Relativistic Fields*. Cambridge University Press, Cambridge, February 1987. ISBN 978-0-521-33707-6.

E. J. G. Pitman. Sufficient statistics and intrinsic accuracy. *Mathematical Proceedings of the Cambridge Philosophical Society*, 32(4):567–579, December 1936. ISSN 1469-8064, 0305-0041. doi: 10.1017/S0305004100019307. URL <https://www.cambridge.org/core/journals/mathematical-proceedings-of-the-cambridge-philosophical-society/article/abs/sufficient-statistics-and-intrinsic-accuracy/6A3E45FB1C423F3F684308F8910D6919>.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1–2):107–136, February 2006. 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. *Information and Inference: A Journal of the IMA*, 8(2):273–288, June 2019. 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, Boston, 4 edition, May 2021. ISBN 978-1-292-40113-3.

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, August 2025a. URL <https://arxiv.org/abs/2508.10096v1>.

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, January 2025b. URL <http://arxiv.org/abs/2501.17913>.

Md Kamruzzaman Sarker, Lu Zhou, Aaron Eberhart, and Pascal Hitzler. Neuro-symbolic artificial intelligence: Current trends. *AI Communications*, 34(3):197–209, March 2022. 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. *Artificial Intelligence*, 70(1):119–165, October 1994. ISSN 0004-3702. doi: 10.1016/0004-3702(94)90105-8. URL <https://www.sciencedirect.com/science/article/pii/0004370294901058>.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates,

Inc., 2017. URL [https://proceedings.neurips.cc/paper\\_files/paper/2017/hash/3f5ee243547dee91fdb053c1c4a845aa-Abstract.html](https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fdb053c1c4a845aa-Abstract.html).

Martin J. Wainwright and Michael Irwin Jordan. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc, 2008. ISBN 978-1-60198-184-4.

Steven R. White. Density-matrix algorithms for quantum renormalization groups. *Physical Review B*, 48(14):10345–10356, October 1993. doi: 10.1103/PhysRevB.48.10345. URL <https://link.aps.org/doi/10.1103/PhysRevB.48.10345>. Publisher: American Physical Society.

## 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 (2009) for Hammersley-Clifford and Casella and Berger (2001) 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 (1971)), we here proof the analogous statement for hypergraphs, where we have to demand the property of clique-capturing defined in Def. 15. We start with showing the following Lemmata to be exploited in the proof.

**Lemma 32** *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  $\tau$  ensures the well-definedness.

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

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

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

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

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

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

Applying the exponential function on both sides establishes the claim.  $\blacksquare$

**Lemma 33** *Let  $\tau$  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*

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

then

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

**Proof** We abbreviate

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

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

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

From the independence assumption it follows that for any index  $x$

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

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

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

Thus, each factor in (8) is trivial, which establishes the claim.  $\blacksquare$

We are finally ready to prove the Hammersley-Clifford Thm. 16 based on the Lemmata above.

**Proof** [Proof of Thm. 16] Need to add the other direction! Restate maybe as a single distribution factorization? By Lem. 32 we have for any index  $x_{\mathcal{V}}$

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

Using the assumption of Thm. 16 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 independent 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. 33 we then have

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

We label by a function

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

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

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

and get, that

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

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

## A.2 Fisher-Neyman

Since sufficient statistics are sometimes introduced based on the data processing inequality (see e.g. Cover and Thomas (2006)), we also show that also that definition is equivalent to the factorization of the family.

**Theorem 34 (Factorization Theorem of Fisher and Neyman)** *Let  $\mathbb{P}$  be a joint distribution of variables  $Z, X$  with values  $\text{val}(Z), \text{val}(X)$  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)  *$Z \rightarrow Y_t \rightarrow X$  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

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

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

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

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

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

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

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

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

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

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

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

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

and with the above at thus at almost all such pairs

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

This is equivalent to *ii)*. ■

Thm. 18 follows from Thm. 34 by the equivalence of *ii)* and *iii)*.