Normal Factor Graphs

by

Ali Al-Bashabsheh

A thesis submitted to the
faculty of graduate and postdoctoral studies
in partial fulfillment of the requirements
for the Ph.D. degree in
electrical engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

This thesis introduces normal factor graphs under a new semantics, namely, the exterior function semantics. Initially, this work was motivated by two distinct lines of research. One line is "holographic algorithms," a powerful approach introduced by Valiant for solving various counting problems in computer science; the other is "normal graphs," an elegant framework proposed by Forney for representing codes defined on graphs. The nonrestrictive normality constraint enables the notion of holographic transformations for normal factor graphs. We establish a theorem, called the generalized Holant theorem, which relates a normal factor graph to its holographic transformation. We show that the generalized Holant theorem on one hand underlies the principle of holographic algorithms, and on the other reduces to a general duality theorem for normal factor graphs, a special case of which was first proved by Forney. As an application beyond Forney's duality, we show that the normal factor graphs duality facilitates the approximation of the partition function for the two-dimensional nearest-neighbor Potts model. In the course of our development, we formalize a new semantics for normal factor graphs, which highlights various linear algebraic properties that enables the use of normal factor graphs as a linear algebraic tool. Indeed, we demonstrate the ability of normal factor graphs to encode several concepts from linear algebra and present normal factor graphs as a generalization of "trace diagrams." We illustrate, with examples, the workings of this framework and how several identities from linear algebra may be obtained using a simple graphical manipulation procedure called "vertex merging/splitting." We also discuss translation association schemes with the aid of normal factor graphs, which we believe provides a simple approach to understanding the subject. Further, under the new semantics, normal factor graphs provide a probabilistic model that unifies several graphical models such as factor graphs, convolutional factor graphs, and cumulative distribution networks.

Acknowledgements

I wish to thank Yongyi Mao for his supervision and friendship. His passion for research is remarkable, and I immensely enjoyed working with him. I am forever grateful to Pascal O. Vontobel and David Forney. Their generous ideas, and several discussions on related topics, have greatly influenced this work. I also wish to thank Frank Kschischang for introducing me to holographic algorithms and early discussions on the subject. I am thankful to Terence H. Chan for discussions on related topics.

I also wish to acknowledge the help and support of my family and friends. Words fail to express my gratitude to Georgios Tzanakis, a true friend. I value the friendship of Lorena Ibacache, José Matos and Anna, and I am thankful to Karen. Last but not least, I am grateful to Julie Dubois for sharing part of the journey, and for her encouragement, support, and understanding.

Contents

List of Figures

Chapter 1

Introduction

1.1 Graphical models

A graphical model is a graphical notation of a mathematical problem, typically, a problem involving the "representation" of a function in terms of "local" functions. One of the main motives for a graphical representation is that it brings together many notions from different disciplines. For instance, factor graphs bridge several topics ranging from statistical physics, machine learning, coding theory, to signal processing. A good example in this direction is the sum-product algorithm, where many algorithms of little historical commonality can be shown to be either equivalent, or instances of the sum-product algorithm. Such list includes the belief propagation from statistics; the BCJR algorithm [4] and the Viterbi algorithm[1] [27, 63] from coding theory; and Kalman filtering from signal processing. The sum-product algorithm remains as the standard decoding algorithm for the capacity approaching codes of low density parity check (LDPC) codes [28] and turbo codes [5]. For an excellent introduction to the interdisciplinary aspect of some graphical models, with focus on factor graphs, we refer the reader to [40].

Below we give a brief introduction to some graphical models from coding theory and probability theory. The field of codes on graphs may be tracked back to the early sixties with the introduction of LDPC codes of Gallager [28]. Later, Tanner introduced what is

[1] The Viterbi algorithm is an instance of the max-product algorithm.

known today as the Tanner graph of a code. A *Tanner graph* [59] is a graph in which each vertex is associated a symbol (digit) or a constraint (subcode), and an edge may only connect a symbol vertex and a constraint vertex, i.e., the graph is bipartite. A symbol vertex and a constraint vertex are connected by an edge if the symbol appears in the constraint. A Tanner graph represents a code consisting of all sequences of symbols (digits) that satisfy all the constraints (subcodes) in the graph. An example Tanner graph is shown in Fig. 1.1. Wiberg et al. [66,67] extended Tanner graphs by introducing states variables, leading to the notion of "generalized state realization" of a code and enabling connections to trellises. A Wiberg-type graph is shown in Fig. 1.2. In [23] Forney introduced *normal realizations*, represented by *normal graphs*, as generalized state realizations in which a state is involved in exactly two constraints and a symbol is involved in one constraint. Due to their importance and intimate relations to this work, we dedicate an introductory section, next section, to normal graphs, and delay their discussion until then.

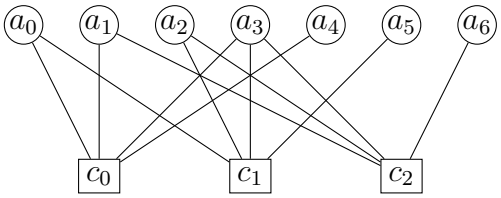


Figure 1.1: An example Tanner graph where $a_0, \ldots, a_6$ are the symbol vertices and c_0, c_1, c_2 are the constraint vertices. Specifying each symbol as a binary symbol, and each constraint (subcode) as the $(4, 3)$ single parity check linear code, one can verify that this Tanner graph represents the $(7, 4)$ Hamming code.

Interpreting each constraint as an indicator function that evaluates to one if the constraint is satisfied and evaluates to zero otherwise, Tanner and Wiberg-type graphs can be viewed as factor graphs. A *factor graph* [38] is a bipartite graph in which each vertex in one of the independent vertex sets is associated a variable, and each vertex from the other independent vertex set is associated a (local) function. An edge connects a variable vertex and a function vertex if the variable is an argument of the function. A factor graph is associated a (global) function, which is defined as the multiplication

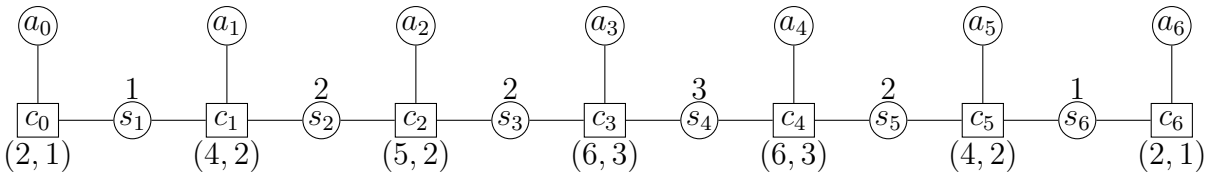


Figure 1.2: A Wiberg-type graph of the $(7,4)$ Hamming code. Each symbol a_i is from the binary field $\mathbb{F}_2$, and the states s_1, s_6 are also from $\mathbb{F}_2$, while $s_2, s_3, s_5 \in \mathbb{F}_2^2$ and $s_4 \in \mathbb{F}_2^3$. For clarity, the dimension (as a vector space over $\mathbb{F}_2$) of each state is listed above its vertex. Each constraint is a linear code, and for illustration we included the length and dimension of each subcode below its corresponding vertex. Note that each state vertex has degree two and each symbol vertex has degree one, i.e., the realization is normal, and so has a natural normal graph representation.

of all the functions in the factor graph. Since their inventions, and equipped with the sum-product algorithm and its abstraction to semirings in general, factor graphs have found enormous applications in many areas ranging from coding theory, signal processing, probability, to statistical physics, and undoubtedly have been revolutionary in some of these areas. A closely related, at least conceptually, graphical model to factor graphs is the model of convolutional factor graphs [44]. A convolutional factor graph is defined in the same way, but unlike a conventional factor graph, its global function is defined as the convolution of all the local functions. That is, while a factor graph describes the multiplicative factorization of a global function, a convolutional factor graph represents the convolutional factorization of the function. Convolutional factor graphs have been used as a probabilistic model to describe the joint distribution of a set of observed random variables that are constructed from independent latent random variables [45], and more recently has been utilized in what is called the linear characteristic model for inference problems with heavy tail distributions [6]. Finally, cumulative distribution networks [30,31], defined as factor graphs whose global function is a cumulative distribution, have found applications in solving some ranking problems.

The graphical models above are related to the probabilistic models of Markov random fields and Bayesian networks [32, 52], and the decoding problem of codes on graphs may

be formulated as an inference problem [37,42,46]. For an excellent and concise discussion of the relation between factor graphs, and Markov random fields and Bayesian networks, see [38]. Below we give a brief description of Bayesian networks and Markov fields, and refer the interested reader to [32, 39, 52] for detailed exposure.

A *Bayesian network* is a directed acyclic graph where each vertex v is associated a random variable X_v. A Bayesian network with a set of vertices V represents a family of distributions that factors as

$$p(X_V) = \prod_{v \in V} p(X_v | X_{\mathrm{pa}(v)}),$$

where $\mathrm{pa}(v)$ is the set of parents of v. An example Bayesian network representing the probability distributions of four random variables that factor as

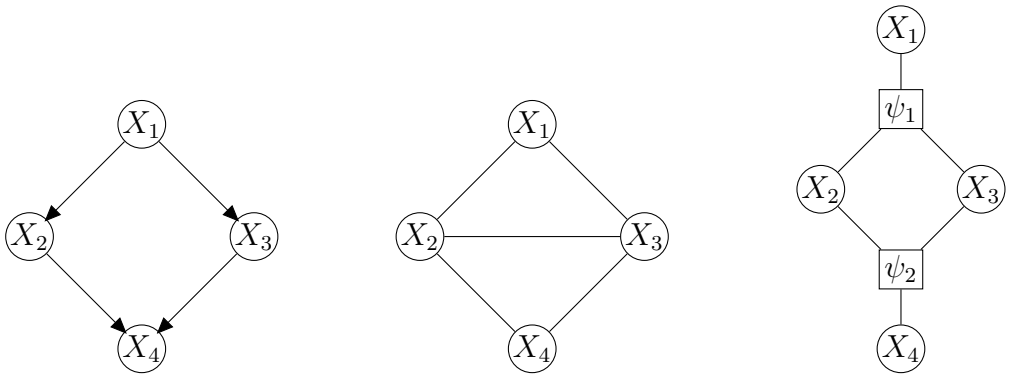
$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2, x_3)$$

is shown in Fig. 1.3 (a). On the other hand, a Markov random field is defined on an undirected graph, where each vertex v is associated a random variable X_v. The random variables satisfy a Markov property such that for any distinct non-adjacent vertices u and v, conditioned on the remaining random variables, X_u and X_v are independent. That is, $X_u \perp\!\!\!\perp X_v | X_{V \setminus \{u,v\}}$, where V is the set of vertices. (This is the pairwise Markov property.) Restricting attention to strictly positive distributions, the Hammersley-Clifford theorem asserts that the joint distribution $p(X_V)$ factors in terms of local functions, called potentials, defined on the maximal cliques[2] of the graph. An example Markov random field representing four random variables with $X_1 \perp\!\!\!\perp X_4 | (X_2, X_3)$ is shown in Fig. 1.3 (b). The family of strictly positive distributions on such random variables must factor as

$$p(x_1, x_2, x_3, x_4) = \psi_1(x_1, x_2, x_3)\psi_2(x_2, x_3, x_4),$$

for some ψ_1 and ψ_2. For illustration, the factor graph representing such factorization is shown in Fig. 1.3 (c).

[2]A clique is a complete subgraph.



(a) Bayesian Network. (b) Markov random field. (c) Factor graph.

Figure 1.3: Probabilistic graphical models.

1.2 Normal factor graphs

The formulation of codes on graphs and the invention of iterative decoding algorithms for graphical codes have undoubtedly revolutionized coding theory. In this research area, the introduction of normal graphs [23] and their duality properties are arguably one of the most elegant and profound results.

In his celebrated paper [23], Forney introduced the notion of *normal realizations*, represented by *normal graphs*, as generalized state realizations of codes. In a normal graph, each vertex represents a local (group) code constraint, and each edge represents a variable that is involved in two constraints at most. A variable involved in two constraints, called a *state* variable, is represented by a regular edge, namely, an edge connecting two vertices; the two vertices correspond to the two constraints involving the variable. A variable involved in only one constraint, called a *symbol* variable, is represented by a "dangling edge,"[3] namely, an edge incident on only one vertex; the vertex corresponds to the single constraint involving the variable. The *global behavior* represented by the normal graph is the set of all symbol-state configurations satisfying all local constraints, and the realized code is the set of all symbol configurations that participate in at least one symbol-state configuration in the global behavior. Forney showed in [23] that codes realized by Tanner graphs or Wiberg-type graphs, in which a variable may in general be

[3] In Forney [23], such edges are called "half-edges".

involved in an unrestricted number of constraints, can be converted to normal graphs by properly replicating some variables.

Normal realizations of codes have a fundamental duality property. By introducing a simple local "dualization" procedure that converts each local code in a normal realization to its dual code and inserts additional "sign inverters," Forney proved a *normal graph duality theorem* [23], which shows that the dualized normal graph realizes the dual code.

The notion of a normal graph may be extended to the notion of a *normal factor graph*, or *Forney-style factor graph* [40,41], which uses an identical graphical representation, but in which the graph vertices no longer represent constraints. Instead, in a normal factor graph, each vertex represents a local *function* involving precisely the variables represented by the edges incident to the vertex. Treated as a factor graph with particular variable-degree restrictions, and interpreted using the standard semantics of factor graphs, a normal factor graph represents a multivariate function that factors as the product of the local function represented by the graph vertices. When each local function in the normal factor graph is the indicator function of a local code constraint, the represented function is the indicator function of the global behavior. This makes normal factor graph representations of codes equivalent to normal graphs, and allows the translation of the normal graph duality theorem to an equivalent theorem for normal factor graphs that represent codes.

Normal factor graphs and Forney's duality theorem have found many applications. For instance, Koetter [34] applied normal graphs (or normal factor graphs for codes) and the duality theorem in a study of trellis formations [36], which gives a necessary and sufficient condition of mergeability and a polynomial-time algorithm for deciding whether a trellis formation contains mergeable vertices. In addition, Koetter *et al.* [35] extended the applications of normal graphs from channel codes to network codes, and used the fundamental duality theorem to establish a reversibility theorem in network coding. Vontobel [64] exploited links between normal factor graphs and electrical circuits and advised a different form of duality (Fenchel's duality). Very recently, Forney used normal factor graphs and Forney's duality theorem to prove a MacWilliams identity for linear codes on graphs in general [25].

1.3 Holographic algorithms

In a seemingly distant research area of complexity theory, Valiant proved, in a groundbreaking work [61], the tractability of some families of combinatorial problems for which no polynomial-time solvers were known previously. In his paper, Valiant developed a very powerful family of algorithms, which he calls *holographic algorithms*, to solve such problems. Holographic algorithms are based on the concept of "holographic reduction," and are governed by a fundamental theorem that Valiant calls the *Holant theorem*.

Although in his original work [61] Valiant dealt only with transforming a product of functions to a specific form, the Holant theorem establishes a principle for transforming an arbitrary product of functions to another product of functions such that the sum over the configuration space is unchanged. Consequently, when computing the sum of a product of functions, the Holant theorem provides a family of transformations that convert the product to a different one, for which the sum may be efficiently computable.

Since many problems in coding and information theory require the computation of sums of products (such as in decoding error correction codes and in computing certain capacities), holographic algorithms become a potentially powerful tool for the information theory community. Indeed, in [58], Schwartz and Bruck showed that certain constrained-coding capacity problems may be solved in polynomial time using holographic algorithms.

1.4 This research

This work stands at the intersection of the above-mentioned lines of research, bridging the areas of Valiant's holographic reduction and Forney's normal graph dualization with the notion of "holographic transformations," a term we coin. The focus of this work will be on general normal factor graphs, in which the vertices may represent arbitrary functions rather than just indicator functions.

In Chapter 2 we introduce a new semantics for normal factor graphs, which we call the "exterior-function semantics." In the exterior-function semantics, instead of letting the graph represent a product of the local functions, we let it represent a sum of products

of local functions, which we call the "exterior function." In this setting, a normal factor graph may be viewed as an expression, or realization, of its exterior function in terms of a "sum of products." In fact, the notion of "sum of products" is the key connection between Forney's normal graph duality theorem and Valiant's Holant theorem: In the case of Valiant [61], this "sum of products" is the number of configurations that needs to be computed, and in the case of Forney [23] (when using normal factor graphs rather than normal graphs to represent codes), this "sum of products" is a code indicator function (possibly up to a scale factor). In this new framework, a holographic transformation is defined as a transformation of a normal factor graph that changes the local functions subject to certain conditions, and converts the normal factor graph to a structurally identical one. The exterior functions of the original normal factor graph and the transformed one are related by what we call the *generalized Holant theorem*.

Chapter 3 is dedicated to the discussion of two applications of the generalized Holant theorem. On one hand, we show that the Holant theorem in [61] is a special case of the generalized Holant theorem, and on the other, we prove a general duality theorem for normal factor graphs as a corollary of the generalized Holant theorem. This duality theorem reduces to Forney's original normal graph duality theorem for normal factor graphs that represent codes. The results in Chapters 2 and 3 appeared in [1].

Another result of our development, addressed in Chapter 4, is a new understanding of normal factor graphs from a linear algebraic perspective. More specifically, the exterior-function semantics associates a normal factor graph with a sum-of-products form, which may be regarded as a linear algebraic expression such as a vector dot product, matrix product, or tensor product. To us, normal factor graphs with the exterior-function semantics appear to be a natural and intuitive language for linear algebra, and may potentially be useful in a variety of applications. Chapter 4 presents several examples on the use of normal factor graphs as a linear algebraic tool, comments on their relation to trace diagrams [56] (a trace diagram reduces to a normal factor graph), and proves a conjecture of Peterson [3, 55, 56]

Chapter 5 presents normal factor graphs as a probabilistic model and discusses their relations to other graphical models, in particular, their relations to factor graphs, con-

volutional factor graphs, and cumulative distribution networks. More specifically, two models are introduced, the constrained and the generative normal factor graphs models, and it is shown that the constrained model is equivalent to factor graphs and a subclass of the generative model is equivalent to convolutional factor graphs. Cumulative distribution networks and the linear characteristic model can be viewed as a transformed generative model.

Recently, Mao and Chan [14] gave a normal factor graph exposition to the Krawtchouk transform and its relation to the Fourier transform and the MacWilliams identities. In Chapter 6 we extend their approach and turn to translation association scheme, a subclass of association schemes [21], and give a normal factor graph discussion of the topic using the notion of convolution from linear-system theory [2].

Very recently, Molkaraie and Loeliger [47] observed empirically that for low temperatures, some stochastic estimators of the partition function of the Ising model converge faster on the dual normal factor than on the primal normal factor graph. In Chapter 7 we provide an analytic and empirical study of such behavior and extend it beyond the Ising model to the Potts model.

The holographic transformations introduced in this work equip normal factor graphs with a rich family of linear transformations, potentially enabling normal factor graphs to serve as a more general analytic framework and computational tool. The power of these transformations, in addition to providing a fundamental duality theorem in coding theory, has also been hinted at by the great power of holographic algorithms (see, *e.g.*, [8–10, 58, 61, 62]).

Chapter 2

Holographic Transformations

2.1 Normal factor graphs: The exterior-function semantics

The term "normal factor graph" has been used in the literature with various meanings. In particular, normal factor graphs as defined in [23] can be easily confused with normal graphs, normal factor graphs for codes, or graphs in which variables are represented by variable vertices. Making a joint effort with Forney [25], we advocate in this work a more rigorous use of the term "normal factor graph," and introduce a new semantics, the "exterior-function" semantics,[1] that defines what a normal factor graph means. To us, this new semantics is quite appealing, since it allows clean development of various graph properties, and has an elegant linear algebraic perspective.

Formally, a *normal factor graph* (NFG) is a graph (V, E), with vertex set V and edge set E, where the edge set E consists of two kinds of edges, a set E^{int} of ordinary edges, each connecting two vertices, and a set E^{ext} of "dangling edges," each having one end attached to a vertex and the other end free.[2] Each edge $e \in E$ represents a

[1] The same semantics is also presented in the concurrent development of Forney [25], in which what we call "exterior functions" are called "partition functions."

[2] More formally, such dangling edges are hyperedges of degree 1, and thus strictly speaking an NFG is a hypergraph rather than a graph.

variable x_e taking values from some finite alphabet $\mathcal{X}_e$; sometimes we may alternatively say that the edge e represents the alphabet $\mathcal{X}_e$ when we do not wish to specify the variable name. Each vertex v represents a complex-valued function[3] f_v on the Cartesian product $\mathcal{X}_{E(v)} := \prod_{e \in E(v)} \mathcal{X}_e$, where $E(v)$ is the set of all edges incident to v. If we denote the set $\{f_v : v \in V\}$ of functions by f_V, then the NFG is specified by the tuple $(V, E^{\text{int}}, E^{\text{ext}}, f_V)$.

When treated as a factor graph under the conventional semantics [38], the NFG $\mathcal{G} = (V, E^{\text{int}}, E^{\text{ext}}, f_V)$ represents the product $\prod_{v \in V} f_v(x_{E(v)})$ of all functions in f_V. This product, expressing a function on $\mathcal{X}_E := \prod_{e \in E} \mathcal{X}_e$, will be called the *interior function*[4] realized by the NFG. Here we have used the standard "variable set" notation $x_{E(v)}$ to denote the set of variables $\{x_e : e \in E(v)\}$.

Now we introduce a new semantics for NFGs: Instead of letting an NFG represent its interior function, we let it represent the interior function summed over all variables represented by regular edges. We call this function the *exterior function* realized by the NFG. More precisely, the exterior function realized by the NFG $\mathcal{G} = (V, E^{\text{int}}, E^{\text{ext}}, f_V)$ is the function

$$Z_{\mathcal{G}}(x_{E^{\text{ext}}}) := \sum_{x_{E^{\text{int}}}} \prod_{v \in V} f_v(x_{E(v)}). \tag{2.1}$$

Thus the exterior function involves only the external variables, represented by the dangling edges. Letting the NFG $\mathcal{G}$ express the function $Z_{\mathcal{G}}$ as defined in (2.1) is what we call the *exterior-function semantics*, which we will use throughout this work.

In this setting, one may view an NFG as an *expression*, or *realization*, of a function (the exterior function) that is given in "sum-of-products" form, as in (2.1), where each variable is involved in either one function or two functions, and the summation is over all variables that are involved in two functions. Since a variable involved in two functions (represented by a regular edge) is "invisible" in the exterior function, we call it an *internal variable*. In contrast, a variable involved only in one function (represented by a dangling edge) remains visible in the exterior function, and is called an *external variable*. An

[3] All functions in this thesis are complex-valued unless we specify otherwise. All results can be generalized to $\mathbb{F}$-valued functions, where $\mathbb{F}$ is an arbitrary field.

[4] In the conventional factor graph literature, an interior function is referred to as a "global function."

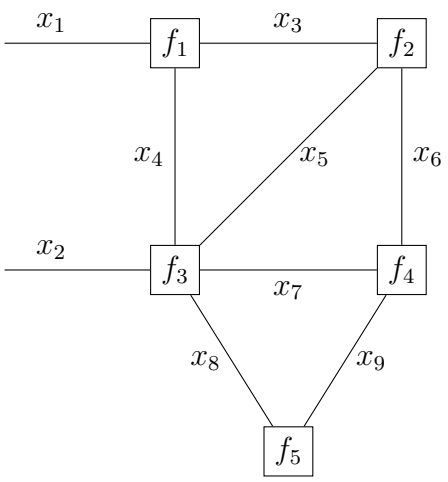


Figure 2.1: A normal factor graph (NFG) $\mathcal{G}$.

example of an NFG is given in Figure 2.1, which realizes the exterior function

$$Z_{\mathcal{G}}(x_1, x_2) = \sum_{x_3,\ldots,x_9} f_1(x_1, x_3, x_4) f_2(x_3, x_5, x_6) f_3(x_2, x_4, x_5, x_7, x_8) f_4(x_6, x_7, x_9) f_5(x_8, x_9).$$

At first glance, NFGs and this semantics may appear to impose a restriction on which sum-of-products forms are representable. We note, however, that *any* sum-of-products form can be straightforwardly converted to one that directly corresponds to an NFG. This requires only that we properly replicate variables, using a "normalization" procedure similar to that of Forney in [23] for converting a factor graph to a normal graph. (Appendix A gives a detailed account of this procedure.) For this reason, using NFGs to represent sum-of-products forms entails no loss of expressive power.

For notational convenience, we may denote the sum-of-products form in (2.1) by[5]

$$\big\langle f_1(x_{E(1)}), f_2(x_{E(2)}), \ldots, f_{|V|}(x_{E(|V|)}) \big\rangle,$$

if V is identified with the set $\{1, 2, \ldots, |V|\}$. Due to the commutativity of both multiplication and summation and the distributive law relating the two operations, it is easy to

[5]In the case of two arguments, say functions f and g, the sum-of-products form $\langle f, g \rangle$ should not be confused with the Hermitian inner product of f and g, whose definition requires complex conjugation of one of the two arguments.

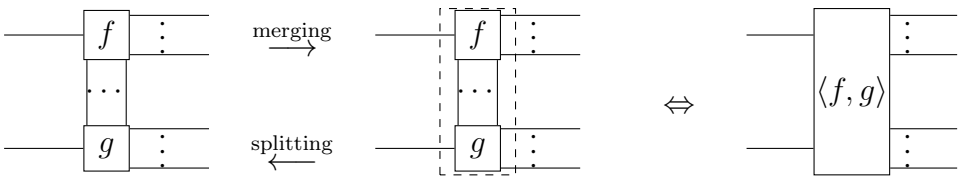


Figure 2.2: Vertex Grouping/Splitting Procedure. Left to right: Vertex Grouping; right to left: Vertex Splitting.

see that any ordering of the arguments of $\langle \cdot, \cdot, \ldots, \cdot \rangle$ expresses the same function. Consequently, we may write the sum-of-products form more compactly as $\langle f_v(x_{E(v)}) : v \in V \rangle$, or even as $\langle f_v : v \in V \rangle$, if no ambiguity results.

2.2 Exterior function preserving procedures

The exterior-function semantics of NFGs allows us to identify immediately several elementary graph manipulation procedures that preserve the exterior function.

Vertex Grouping/Splitting Procedure In a Vertex Grouping Procedure, two vertices representing functions f and g are grouped together, and the group is replaced by a vertex representing the function $\langle f, g \rangle$. In a Vertex Splitting Procedure, a vertex representing a function that can be expressed by the sum-of-products form $\langle f, g \rangle$ is replaced by an NFG representing $\langle f, g \rangle$. Figure 2.2 shows this pair of procedures.

Lemma 1 *Applying a Vertex Grouping or Vertex Splitting Procedure to an NFG preserves the realized exterior function.*

Proof: This lemma holds because these procedures simply correspond to conversions between sum-of-products forms $\langle f, g, h_1, \ldots, h_m \rangle$ and $\big\langle \langle f, g \rangle, h_1, h_2, \ldots, h_m \big\rangle$, which evidently express the same function. □

We note that this pair of procedures were first introduced by Loeliger in [40,41], who refers to Vertex Grouping as "closing the box," and to Vertex Splitting as "opening the box." However, in [40,41] these procedures are used to interpret an NFG (in the original

semantics) as a flexible hierarchical model, and to explain message-passing algorithms, rather than in the context of the exterior-function semantics.

Note that the Vertex Grouping Procedure may be applied recursively to an arbitrary number of vertices, say $f_1, f_2, \ldots, f_m$, so that these functions are replaced by a single function realized by $\langle f_1, f_2, \ldots, f_m \rangle$. The resulting NFG still realizes the same exterior function. Similarly, the reverse Vertex Splitting Procedure also preserves the exterior function. For these reasons, when we draw a dashed box (as in Figure 2.2, middle) to group some vertices, we may freely interpret the NFG as the equivalent NFG in which the box is replaced by a single vertex representing the function realized by the box. (Figure 2.2, right.)

Equality Insertion/Deletion Procedure For any finite alphabet $\mathcal{X}$, let $\delta_=$ denote the $\{0,1\}$-valued function on $\mathcal{X} \times \mathcal{X}$ which evaluates to 1 if and only if the two arguments of the function are equal. That is, $\delta_=$ is an "equality indicator function." In an Equality Insertion Procedure, a $\delta_=$ function is inserted into an edge; in an Equality Deletion Procedure, a $\delta_=$ is deleted and the two edges originally connected to the function are joined. Figure 2.3 shows this pair of procedures.

Figure 2.3: Equality Insertion/Deletion Procedure. Left to right: Equality Insertion; right to left: Equality Deletion. The edges may be regular or dangling; the vertex labelled with "=" represents the function $\delta_=$.

Lemma 2 *Applying an Equality Insertion or Deletion Procedure to an NFG preserves the realized exterior function.*

Proof: This result is a corollary of Lemma 1, and simply follows from the fact that if the $\delta_=$ function is inserted into an edge incident to a function f, then we may group f with $\delta_=$ and replace the sum-of-products form $\langle f, \delta_= \rangle$ with the function it expresses, namely f. □

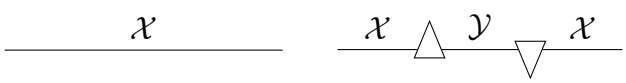


Figure 2.4: Dual Vertex Insertion/Deletion Procedure. Left to right: Dual Vertex Insertion; right to left: Dual Vertex Deletion. The edges may be regular or dangling; the oppositely oriented triangular vertices represent a dual pair of functions.

Dual Vertex Insertion/Deletion Procedure Let $\mathcal{X}$ and $\mathcal{Y}$ be two finite alphabets and $\Phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{C}$ and $\widehat{\Phi} : \mathcal{X} \times \mathcal{Y} \to \mathbb{C}$ be two functions. Then we say that Φ and $\widehat{\Phi}$ are *dual* with respect to the alphabet $\mathcal{Y}$, and call $\mathcal{Y}$ the *coupling* alphabet, if $\left\langle \Phi(x,y), \widehat{\Phi}(x',y) \right\rangle = \delta_=(x,x')$ for every $x, x' \in \mathcal{X}$. In the case when $\mathcal{X}$ and $\mathcal{Y}$ have the same cardinality, we also call the vertices representing Φ and $\widehat{\Phi}$ *transformers*, a term which will be justified in Section 2.3. In a Dual Vertex Insertion Procedure, we insert into an edge representing alphabet $\mathcal{X}$ a dual pair of functions $\Phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{C}$ and $\widehat{\Phi} : \mathcal{X} \times \mathcal{Y} \to \mathbb{C}$, with $\mathcal{Y}$ being the coupling alphabet, and let the edge connecting the two functions represent $\mathcal{Y}$, Fig. 2.4. A Dual Vertex Deletion Procedure is the reverse of a Dual Vertex Insertion Procedure, in which we delete a pair of dual functions and the edge connecting them, and then join the ends of the two cut edges, Fig. 2.4.

Lemma 3 *Applying a Dual Vertex Insertion or Deletion Procedure to an NFG preserves the realized exterior function.*

Proof: The Dual Vertex Insertion Procedure is equivalent to first inserting a $\delta_=$ function in the edge and then splitting the $\delta_=$ function into a pair of dual functions. The lemma then follows from Lemmas 1 and 2. □

2.3 A linear algebraic perspective

Before we proceed to introduce holographic transformations, we pause to interpret NFGs from a linear algebraic perspective. A more elaborate encounter with linear algebra is provided in Chapter 4.

We will denote the set of all complex-valued functions on a finite alphabet $\mathcal{X}$ by $\mathbb{C}^{\mathcal{X}}$. It is well known that $\mathbb{C}^{\mathcal{X}}$ is isomorphic to the vector space $\mathbb{C}^{|\mathcal{X}|}$: after imposing

an order on $\mathcal{X}$, one can arrange the values of any function $f \in \mathbb{C}^{\mathcal{X}}$ as a vector in $\mathbb{C}^{|\mathcal{X}|}$ according to that order. Similarly, depending on the structure of $\mathcal{X}$, the function f may also be viewed as a matrix, or as its higher-dimensional generalization, namely a tensor; if $\mathcal{X}$ is the Cartesian product $\mathcal{X}_1 \times \mathcal{X}_2$ of some alphabets $\mathcal{X}_1$ and $\mathcal{X}_2$, then f may be regarded as a matrix; if $\mathcal{X}$ is a multifold Cartesian product of alphabets, then f may be viewed as a multi-dimensional array, or as a tensor. On the other hand, conventional linear algebraic objects like vectors, matrices and tensors may be alternatively regarded as multivariate functions. In particular, a tensor with n indices may be identified with a multivariate function involving n variables. From this perspective, any sum-of-products form corresponding to an NFG may be viewed as a linear algebraic expression.

In the simplest case, Fig. 2.5 shows the NFG realizing the sum-of-products form $\big\langle f(x_I), g(x_J) \big\rangle$ involving exactly two functions $f : \mathcal{X}_I \to \mathbb{C}$ and $g : \mathcal{X}_J \to \mathbb{C}$, where I and J are two finite index sets, possibly intersecting, and where for every $i \in I \cup J$, $\mathcal{X}_i$ is an arbitrary finite alphabet. In this figure each set of variables is treated as a single variable and hence represented by a single edge. If such a set is empty, the corresponding edge (or dangling edge) simply disappears from the figure. From the definition of the exterior function, we have

$$\langle f, g \rangle = \sum_{x_{I \cap J}} f(x_I) g(x_J),$$

and it is straightforward to verify the following propositions:

- If $I = J \neq \emptyset$, then $\langle f, g \rangle$ is the dot product $f \cdot g$, where f and g are regarded as $|\mathcal{X}_I|$-dimensional vectors and "$\cdot$" is a dot product.

- If $I \supset J \neq \emptyset$, then $\langle f, g \rangle$ is the matrix-vector product $f \cdot g$, where f is regarded as a $|\mathcal{X}_{I \setminus J}| \times |\mathcal{X}_J|$ matrix, g is regarded as a $|\mathcal{X}_J|$-dimensional vector, and "$\cdot$" is a matrix-vector product.

- If $I \setminus J$, $J \setminus I$ and $I \cap J$ are all non-empty, then $\langle f, g \rangle$ is the matrix-matrix product $f \cdot g$, where f is regarded as a $|\mathcal{X}_{I \setminus J}| \times |\mathcal{X}_{I \cap J}|$ matrix, g is regarded as $|\mathcal{X}_{I \cap J}| \times |\mathcal{X}_{J \setminus I}|$ matrix, and "$\cdot$" is a matrix-matrix product.

- If I and J are disjoint and both non-empty, then $\langle f, g\rangle$ is the vector outer product, matrix Kronecker product, or tensor product, $f \cdot g$, where f and g are regarded as two vectors, two matrices, or two tensors, respectively, and "$\cdot$" is the corresponding product operation.

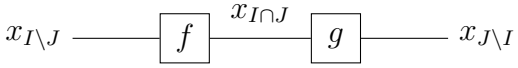


Figure 2.5: The NFG $\mathcal{G}$ representing the simple sum-of-products form $\langle f|g\rangle$ where $I\setminus J = \{i \in I | i \notin J\}$ and $J\setminus I = \{j \in J | j \notin I\}$.

In summary, this simple sum-of-products form, namely $\langle f, g\rangle$, unifies various notions of "product" in linear algebra. This unification illustrates the convenience of understanding linear algebraic objects such as vectors, matrices and tensors as multivariate functions, since in this perspective one never needs to be concerned with whether a vector is a row or column vector, whether a matrix is transposed, and so forth.

A general sum-of-products form which involves multiple functions and which can be represented by an NFG may be viewed as a linear algebraic expression involving various such linear algebraic objects and various such notions of product. The fact that the sum-of-products form $\langle \cdot, \cdot, \ldots, \cdot\rangle$ does not depend on how its arguments are ordered contrasts with the standard order-dependent notations in linear algebra.

In this perspective, Lemma 1 follows from the order-independent nature of sum-of-products forms, and Lemma 2 follows from the fact that $\delta_=$ is essentially an identity matrix.

In linear algebra, vectors, matrices and tensors may be viewed alternatively as linear maps, which are characterized by the spaces they act on and the product operation used in defining the maps. Similar perspectives can be made explicit in the NFG context. For example, a complex-valued bivariate function $f(x, y)$ defined on $\mathcal{X} \times \mathcal{Y}$ may be viewed as two maps: When participating in the sum-of-products form $\big\langle f(x, y), g(y)\big\rangle$ with a function $g : \mathcal{Y} \to \mathbb{C}$, f can be viewed as a linear map from the vector space $\mathbb{C}^{\mathcal{Y}}$ to the vector space $\mathbb{C}^{\mathcal{X}}$; when participating in the sum-of-products form $\big\langle f(x, y), h(x)\big\rangle$ with a

function $h : \mathcal{X} \to \mathbb{C}$, f can be viewed as a linear map from the vector space $\mathbb{C}^{\mathcal{X}}$ to the vector space $\mathbb{C}^{\mathcal{Y}}$.

This aspect allows us to interpret dual functions in two different ways. Suppose that $\Phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{C}$ and $\widehat{\Phi} : \mathcal{X} \times \mathcal{Y} \to \mathbb{C}$ are a pair of dual functions. On one hand, we may view Φ as a map from $\mathbb{C}^{\mathcal{X}}$ to $\mathbb{C}^{\mathcal{Y}}$ and $\widehat{\Phi}$ as a map from $\mathbb{C}^{\mathcal{Y}}$ back to $\mathbb{C}^{\mathcal{X}}$. In this view, the composition map $\widehat{\Phi} \circ \Phi$ is the identity map from $\mathbb{C}^{\mathcal{X}}$ to $\mathbb{C}^{\mathcal{X}}$, and $\widehat{\Phi}$ is essentially the inverse or pseudo-inverse of Φ. On the other hand, we may view Φ as a map from $\mathbb{C}^{\mathcal{X}}$ to $\mathbb{C}^{\mathcal{Y}}$ and $\widehat{\Phi}$ also as a map from $\mathbb{C}^{\mathcal{X}}$ to $\mathbb{C}^{\mathcal{Y}}$. In this view, the dot product of two vectors f and g in $\mathbb{C}^{\mathcal{X}}$ is preserved after they are mapped, respectively, to vectors $\langle f, \Phi \rangle$ and $\langle g, \widehat{\Phi} \rangle$ in $\mathbb{C}^{\mathcal{Y}}$. This view is justified by $\langle f, g \rangle = \left\langle f, \Phi, \widehat{\Phi}, g \right\rangle = \left\langle \langle f, \Phi \rangle, \langle \widehat{\Phi}, g \rangle \right\rangle$.

Finally, we justify the term "transformer" that was introduced in Section 2.2. Suppose that the functions Φ and $\widehat{\Phi}$ on $\mathcal{X} \times \mathcal{Y}$ are dual with respect to alphabet $\mathcal{Y}$, and that $\mathcal{X}$ and $\mathcal{Y}$ have the same cardinality. Then we may identify Φ with its square-matrix representation in which the rows are indexed by $\mathcal{X}$ and the columns are indexed by $\mathcal{Y}$; similarly, we may identify $\widehat{\Phi}$ with its square-matrix representation in which the rows are indexed by $\mathcal{Y}$ and the columns are indexed by $\mathcal{X}$. The fact that Φ and $\widehat{\Phi}$ are dual with respect to $\mathcal{Y}$ implies that the matrix product $\Phi \cdot \widehat{\Phi}^T$ is the identity matrix. Thus the matrix $\widehat{\Phi}$ is the unique inverse of the matrix Φ, and *vice versa*. Therefore, the functions Φ and $\widehat{\Phi}$ may be regarded as a pair of *transformations* (or transformation kernels) that are inverse to each other.

2.4 Holographic transformations and the generalized Holant theorem

Now we are ready to define holographic transformations.

Suppose that I is a finite index set and that for each $i \in I$, there are two finite alphabets $\mathcal{X}_i$ and $\mathcal{Y}_i$ having the same cardinality. We will call a function $\Phi : \mathcal{X}_I \times \mathcal{Y}_I \to \mathbb{C}$ a *separable transformation* if Φ is a transformation from $\mathbb{C}^{\mathcal{X}_I}$ to $\mathbb{C}^{\mathcal{Y}_I}$ (namely, there exists a unique function $\widehat{\Phi} : \mathcal{X}_I \times \mathcal{Y}_I \to \mathbb{C}$ such that $\langle \Phi(x, y), \widehat{\Phi}(x', y) \rangle = \delta_=(x, x')$ for

all $x, x' \in \mathcal{X}$), and there exists a collection of functions $\{\Phi_i \in \mathbb{C}^{\mathcal{X}_i \times \mathcal{Y}_i} : i \in I\}$ such that $\Phi = \prod_{i \in I} \Phi_i$. Noting that $\prod_{i \in I} \Phi_i$ may be identified with the sum-of-products form $\langle \Phi_i : i \in I \rangle$, we see that transforming any function in $f \in \mathbb{C}^{\mathcal{X}_I}$ by Φ is equivalent to evaluating the sum-of-products form $\big\langle f, \langle \Phi_i : i \in I \rangle \big\rangle$, which can be performed via *separately* transforming f by each of the Φ_i's in an arbitrary order; hence the term "separable."

It is easy to verify that if $\Phi := \prod_{i \in I} \Phi_i$ is a separable transformation, then its inverse transformation $\widehat{\Phi} : \mathcal{X}_I \times \mathcal{Y}_I \to \mathbb{C}$ is $\prod_{i \in I} \widehat{\Phi}_i$ where each $\widehat{\Phi}_i : \mathcal{X}_i \times \mathcal{Y}_i \to \mathbb{C}$ is the inverse transformation of Φ_i. It follows that if a transformation is separable, then so is its inverse.

Holographic Transformation Let $\mathcal{G} = (V, E^{\text{int}}, E^{\text{ext}}, f_V)$ be an NFG where for each edge $e \in E := E^{\text{ext}} \cup E^{\text{int}}$, $\mathcal{X}_e$ is the alphabet of the represented variable. For each $e \in E$, let $\mathcal{Y}_e$ be another alphabet having the same cardinality as $\mathcal{X}_e$. For every vertex v and every edge $e \in E(v)$, we associate a transformation $\Phi_{v,e} : \mathcal{X}_e \times \mathcal{Y}_e \to \mathbb{C}$ such that if e is a regular edge connecting vertices u and v, then $\Phi_{u,e}$ and $\Phi_{v,e}$ are the inverse transformations of each other. Let $\Phi_v := \prod_{e \in E(v)} \Phi_{v,e}$ for every vertex v. Locally transform each function f_v to the function $F_v \in \mathbb{C}^{\mathcal{Y}_{E(v)}}$ via $F_v := \langle f_v, \Phi_v \rangle$, and collectively denote $\{F_v : v \in V\}$ by F_V. We call the NFG $\mathcal{G}^H := (V, E^{\text{int}}, E^{\text{ext}}, F_V)$ the holographic transformation of $\mathcal{G}$ with respect to the collection of local separable transformations $\{\Phi_v : v \in V\}$.

A graphical example of holographic transformation is shown in Figure 2.6. We note that holographic transformations keep the topology of the NFG unchanged, and only transform each local function.

Theorem 1 (Generalized Holant Theorem) *In the setting above, the exterior function $Z_{\mathcal{G}^H}$ of the NFG $\mathcal{G}^H$ is related to the exterior function $Z_{\mathcal{G}}$ of the original NFG $\mathcal{G}$ by*

$$Z_{\mathcal{G}^H}(y_{E^{\text{ext}}}) = \big\langle Z_{\mathcal{G}}(x_{E^{\text{ext}}}), \langle \Phi_e(x_e, y_e) : e \in E^{\text{ext}} \rangle \big\rangle,$$

where for each $e \in E^{\text{ext}}$, we have written Φ_e in place of $\Phi_{v,e}$.

Proof: The theorem simply follows from Lemma 3. Graphically, as shown in Figure 2.6,

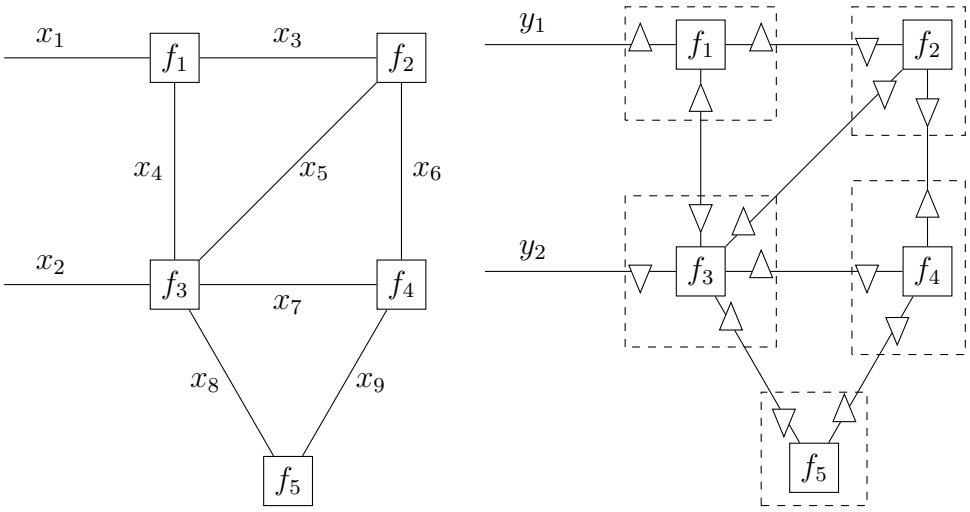


Figure 2.6: Holographic transformation: an NFG $\mathcal{G}$ (left) and its holographic transformation $\mathcal{G}^H$ (right). Each triangular vertex is a transformer, possibly different. Oppositely oriented transformers on an edge are the inverses of each other.

the holographic transformation is equivalent to first inserting into each edge an inverse pair of transformers and into each dangling edge a transformer, and then transforming each local function by its surrounding transformers. Since each inverse pair of transformers cancels out, the only difference between the exterior function of $\mathcal{G}$ and that of $\mathcal{G}^H$ is due to the transformers that have been inserted in the dangling edges. This establishes the theorem. □

When E^{ext} is the empty set, the exterior functions of $\mathcal{G}$ and $\mathcal{G}^H$ reduce to scalars. In this case, the generalized Holant theorem reduces to the Holant theorem of [61].

We note also that in the literature of "loop calculus" [15–18], which has been introduced for the study of belief propagation and of the partition functions of statistical mechanics models, a result equivalent to the Holant theorem has been proved, and a transformation equivalent to our holographic transformation (on NFGs without dangling edges) has been proposed under the name of "gauge transformation" (see, *e.g.*, [16]).

Although our generalization of the Holant theorem appears straightforward, we believe that there is a conceptual leap in this generalization. In particular, the original Holant theorem reveals only that there are redundant and structurally identical NFGs

that may be used to represent the same scalar quantity as a sum of products; it makes no attempt to transform or reparameterize an exterior function that assumes a more general form. In the general setting of holographic transformations, when the original NFG $\mathcal{G}$ is viewed as a realization of some function $g := Z_{\mathcal{G}}$ on a collection of alphabets $\{\mathcal{X}_e : e \in E\}$, the holographically transformed NFG $\mathcal{G}^H$ is viewed as a realization of a related function $g^H := Z_{\mathcal{G}^H}$ on a different collection of alphabets $\{\mathcal{Y}_e : e \in E\}$. In particular, the function g^H may be regarded as a transform-domain representation of g via an "external change of basis;" namely, a change of basis for the vector space $\mathbb{C}^{|\mathcal{X}_{E^{\mathrm{ext}}}|}$, that is characterized by the "external transformation" $\langle \Phi_e : e \in E^{\mathrm{ext}} \rangle$, where this "external change of basis" involves, in its sum-of-products form, a "local change of basis" for each component vector space $\mathbb{C}^{|\mathcal{X}_e|}, e \in E$.

Chapter 3

Applications of Holographic Transformations

In this chapter we discuss the normal graph duality of Forney [23, 44], and the holographic transformations of Valiant [61]. We show that, despite their seemingly distant nature, both Forney's duality theorem and Valiant's Holant theorem are instances of the GHT. More applications of holographic transformations are presented in Chapter 5 in the context of probabilistic models.

3.1 Duality in normal factor graphs

The first duality theorem for codes on graphs was the normal graph duality theorem that was introduced by Forney in [23]. In the setting of [23], the graphs considered, rather than being NFGs, are "normal graphs," where edges incident on one or two vertices represent "symbol" variables and "state" variables, respectively, and where each vertex represents a local group-code constraint. The global behavior represented by the graph is the set of all symbol-state configurations that satisfy all local constraints, and the graph itself represents a group code that consists of all symbol configurations that participate in at least one symbol-state configuration in the global behavior. In [23], Forney introduced a local "dualization" procedure for normal graphs, which converts

each local code constraint to its dual code constraint and inserts a "sign inverter" into every edge connecting two vertices. The normal graph duality theorem then states that the dualized graph represents the dual code. The normal graph duality theorem of [23] may be formulated as an equivalent theorem, which we call the "code normal factor graph duality theorem," using the language of normal factor graphs. More specifically, we may use an NFG to represent a state realization of a group code C, where each vertex represents the indicator function of a local group code constraint, and the exterior function is, up to scale,[1] the indicator function of the code C. The dualization procedure may be reformulated on the NFG as converting the indicator function of each local code to the indicator function of the dual of the local code and inserting an indicator function δ_+ into each edge, where δ_+ evaluates to 1 if and only if the two arguments of the function are additive inverses of each other. Then the code normal factor graph duality theorem states that the exterior function realized by the dual NFG is up to scale the indicator function of the dual code $C^\perp$.

In the framework of factor graphs, Mao and Kschischang [44] introduced the notions of multivariate convolution and convolutional factor graphs, and proved a duality theorem between a multiplicative factor graph and its dual convolutional factor graph. The duality theorem of [44] (Theorem 11), which we call the "MK theorem," states that a dual pair of factor graphs represent a Fourier transform pair. Since the indicator function of a code and that of its dual code are a Fourier transform pair up to scale, the code normal graph duality theorem and hence the normal graph duality theorem follow from the MK theorem as corollaries.

In a concurrent development [25], Forney has established a general normal factor graph duality theorem, where the vertices of an NFG can represent arbitrary functions and the dualization procedure is defined as converting each local function to its Fourier transform and inserting a δ_+ function into each edge. The general normal factor graph duality theorem states that a dual pair of NFG's represent a Fourier transform pair up

[1]The scaling constant is the number of symbol-state configurations in the global behavior that correspond to each codeword; this number is the same for every codeword since the global behavior is an abelian group.

to scale. This theorem reduces to the code normal factor graph duality theorem (and hence to the normal graph duality theorem) if each graph vertex is the indicator function of a local code.

In this section, we will show that the general normal factor graph duality theorem follows directly from the generalized Holant theorem.

Let $\mathcal{G} = (V, E^{\text{int}}, E^{\text{ext}}, f_V)$ be an arbitrary NFG, where each variable alphabet $\mathcal{X}_e, e \in E = E^{\text{ext}} \cup E^{\text{int}}$, is a finite abelian group written additively. It is well-known that every finite abelian group $(\mathcal{X}, +)$ has a character group $\mathcal{X}^\wedge$, consisting of precisely the set of all homomorphisms, called characters, of $\mathcal{X}$ mapping $\mathcal{X}$ into the multiplicative group of the unit circle in the complex plane. The character group $\mathcal{X}^\wedge$ of $\mathcal{X}$ has the following properties [22].

- The group operation $+$ in $\mathcal{X}^\wedge$ is defined by $(\hat{x}_1 + \hat{x}_2)(x) = \hat{x}_1(x)\hat{x}_2(x)$ for any two characters $\hat{x}_1, \hat{x}_2 \in \mathcal{X}^\wedge$ and any $x \in \mathcal{X}$.

- $(\mathcal{X}^\wedge)^\wedge$ is isomorphic to $\mathcal{X}$. This result, known as Pontryagin duality [22], allows each element of $\mathcal{X}$ to be treated as a character of $\mathcal{X}^\wedge$.

- $\mathcal{X}^\wedge$ is isomorphic to $\mathcal{X}$.

- For each $x \in \mathcal{X}$ and $\hat{x} \in \mathcal{X}^\wedge$, $x(\hat{x}) = \hat{x}(x)$. We will denote[2] both $x(\hat{x})$ and $\hat{x}(x)$ by $\kappa_{\mathcal{X}}(x, \hat{x})$ and, for later use, denote $\kappa_{\mathcal{X}}(x, -\hat{x})/|\mathcal{X}|$ by $\widehat{\kappa}_{\mathcal{X}}(x, \hat{x})$. Keeping in mind that $\kappa_{\mathcal{X}}$ and $\widehat{\kappa}_{\mathcal{X}}$ are both defined with respect to the alphabet $\mathcal{X}$, we may sometimes suppress such dependency in our notation. It is easy to see that $\kappa_{\mathcal{X}}$ and $\widehat{\kappa}_{\mathcal{X}}$ are a dual pair of functions (with respect to either $\mathcal{X}$ or $\mathcal{X}^\wedge$). Since $\mathcal{X}$ and $\mathcal{X}^\wedge$ have the same size, they in fact define a pair of transformations, namely, the Fourier transform and its inverse, as we state next.

- For any function $f \in \mathbb{C}^{\mathcal{X}}$, its Fourier transform $\mathcal{F}[f]$ is a complex-valued function on $\mathcal{X}^\wedge$ defined by $\mathcal{F}[f] := \langle f, \kappa \rangle$. It follows that for any function $f \in \mathbb{C}^{\mathcal{X}^\wedge}$, its inverse Fourier transform $\mathcal{F}^{-1}[f]$ is a complex-valued function on $\mathcal{X}$, $\mathcal{F}^{-1}[f] = \langle f, \widehat{\kappa} \rangle$. We

[2]It is customary in the literature to denote both $x(\hat{x})$ and $\hat{x}(x)$ by the pairing $\langle x, \hat{x} \rangle$. But we choose not to use this notation since it collides with our notation for "sum-of-products" forms.

note that the inverse Fourier transform operator $\mathcal{F}^{-1}$ may also be applied to a function $f \in \mathbb{C}^{\mathcal{X}}$ and result in a function on $\mathcal{X}^\wedge$, where in the sum-of-products form the summation is over the $\mathcal{X}$-valued variable.

- If $\mathcal{X}$ is the direct product $\mathcal{X}_1 \times \mathcal{X}_2$ of finite abelian groups $\mathcal{X}_1$ and $\mathcal{X}_2$, then $\mathcal{X}^\wedge$ is the direct product $\mathcal{X}_1^\wedge \times \mathcal{X}_2^\wedge$ of the character groups $\mathcal{X}_1^\wedge$ and $\mathcal{X}_2^\wedge$. In this case, for any $(x_1, x_2) \in \mathcal{X}_1 \times \mathcal{X}_2$ and any $(\hat{x}_1, \hat{x}_2) \in \mathcal{X}_1^\wedge \times \mathcal{X}_2^\wedge$,

$$\kappa\big((x_1, x_2), (\hat{x}_1, \hat{x}_2)\big) = \kappa(x_1, \hat{x}_1)\kappa(x_2, \hat{x}_2). \tag{3.1}$$

  Equation (3.1) states that the Fourier transform is a separable transformation.

Returning to the NFG $\mathcal{G}$, we next obtain its "dual" by applying Forney's dualization procedure.

NFG Dualization Procedure Replace each $\mathcal{X}_e$-valued variable x_e by an $\mathcal{X}_e^\wedge$-valued variable $\hat{x}_e$. Replace each function $f_v, v \in V$, by its Fourier transform $\mathcal{F}[f_v]$ and insert into each internal edge a vertex representing the function $\delta_+(\cdot)$. Again, the function δ_+ is an indicator function which evaluates to 1 if and only if its two variables are the additive inverses of each other. We will denote the resulting NFG by $\widehat{\mathcal{G}}$, and refer to it as the *dual* NFG of $\mathcal{G}$.

Theorem 2 (General NFG Duality Theorem) *The exterior function $Z_{\widehat{\mathcal{G}}}$ realized by the dual NFG $\widehat{\mathcal{G}}$ and the exterior function $Z_{\mathcal{G}}$ realized by the original NFG $\mathcal{G}$ are related by*

$$Z_{\widehat{\mathcal{G}}} = |\mathcal{X}_{E^{\text{int}}}| \cdot \mathcal{F}[Z_{\mathcal{G}}]. \tag{3.2}$$

Proof: This theorem can be viewed as a corollary of the generalized Holant theorem, and can be simply proved graphically (Figure 3.1): Given $\mathcal{G}$, construct another NFG $\mathcal{G}'$ by inserting a $\delta_=$ function into each regular edge $\mathcal{G}$; by Lemma 2, $Z_{\mathcal{G}'} = Z_{\mathcal{G}}$. Obtain the NFG ${\mathcal{G}'}^H$ from $\mathcal{G}'$ by inverse Fourier transforming every $\delta_=$ in $\mathcal{G}'$ and Fourier transforming every other function. This corresponds to inserting the dual functions $\kappa_{\mathcal{X}_e}$ and $\widehat{\kappa}_{\mathcal{X}_e}$ into each regular edge e (with $\widehat{\kappa}_{\mathcal{X}_e}$ adjacent to the function $\delta_=$) and inserting the function $\kappa_{\mathcal{X}_e}$

into each dangling edge e. Since the inserted transformers in each regular edge are the inverses of each other, this verifies that ${\mathcal{G}'}^H$ is a holographic transformation of $\mathcal{G}'$. By the generalized Holant theorem,

$$Z_{{\mathcal{G}'}^H} = \big\langle Z_{\mathcal{G}'}, \langle \kappa_e : e \in E^{\text{ext}} \rangle \big\rangle = \big\langle Z_{\mathcal{G}}, \langle \kappa_e : e \in E^{\text{ext}} \rangle \big\rangle = \mathcal{F}[Z_{\mathcal{G}}].$$

Invoking a well-known result that for $\delta_=$ defined on $\mathcal{X} \times \mathcal{X}$, $\mathcal{F}^{-1}[\delta_=] = \frac{1}{|\mathcal{X}|}\delta_+$, we see that ${\mathcal{G}'}^H$ and $\widehat{\mathcal{G}}$ are in fact identical except that in ${\mathcal{G}'}^H$, each δ_+ inserted in edge e is scaled by $\frac{1}{|\mathcal{X}_e|}$. The theorem is then proved by collecting all the scaling factors. □

We note that Theorem 2 is the most general NFG duality theorem. If each function in an NFG is an indicator function of a local code, then the exterior function realized by the NFG is up to scale the indicator function of a group code. Such an NFG, which may be called a "code normal factor graph" ("code NFG") may then be used to represent the group code. This makes a code NFG equivalent to a normal graph. Since the indicator function of a code and that of its dual are (up to scale) a Fourier transform pair, the general normal factor graph duality theorem then reduces to the code normal graph duality theorem (and the normal graph duality theorem), which state that a dual pair of code NFGs (resp. a dual pair of normal graphs) represent a pair of dual codes.

It is worth noting that the general normal factor graph duality theorem also follows from the MK theorem, via the application of the "projection-slice theorem" of the Fourier transform, or the "sampling/averaging duality theorem" of [44, Theorem 8]. However, the proof using the generalized Holant theorem seems more transparent.

3.2 Holographic reduction

A holographic reduction may be regarded as a particular kind of holographic transformation applied to an NFG without dangling edges, by which a counting problem may be reduced to an equivalent "PerfMatch problem". Using this technique, Valiant constructed polynomial-time solvers for various families of "counting" problems previously unknown to be in P [61]. We now summarize the main results of Valiant [61] and explain how holographic reduction works.

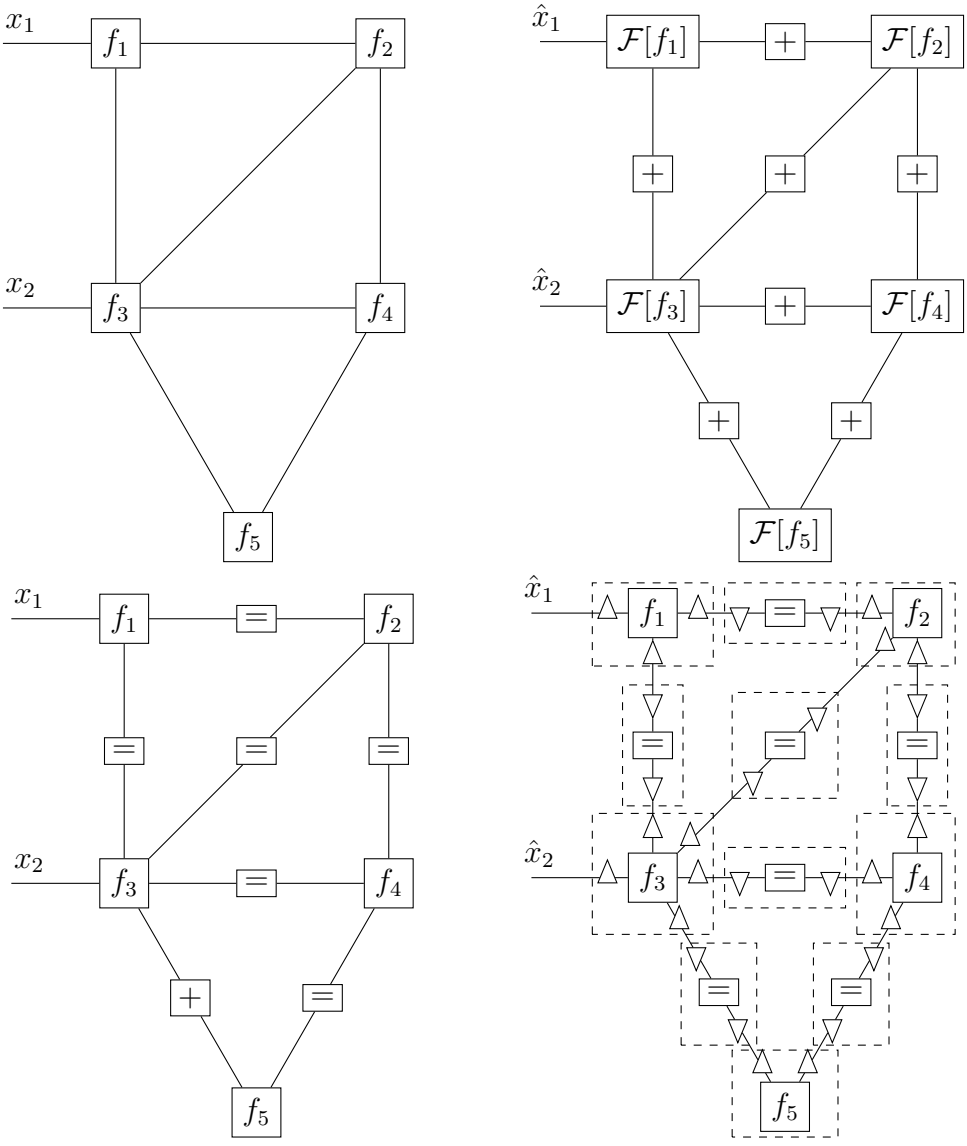


Figure 3.1: NFG's $\mathcal{G}$ (top left), $\widehat{\mathcal{G}}$ (top right), $\mathcal{G}'$ (bottom left) and $\mathcal{G}'^H$ (bottom right).

The PerfMatch Problem Suppose that $H = (V, E, w)$ is a weighted graph with vertex set V, edge set E, and weighting function w which assigns to each edge $e \in E$ a complex weight $w(e)$. The quantity PerfMatch $\pi(H)$ of H is defined as

$$\pi(H) := \sum_{M \in Q(H)} \prod_{e \in M} w(e),$$

where $Q(H)$ is the collection of all perfect matchings[3] of H. It is known that the PerfMatch problem, namely, solving for $\pi(H)$, can be performed in polynomial time using the FKT algorithm [33, 60] if H is a *planar* graph.

A principle underlying holographic reduction is a graph-theoretic property which expresses the PerfMatch of a weighted graph as a sum-of-products form, in which each involved function is defined based on a local component of the graph. Such a local component is referred to as a "matchgate," and each involved function is referred to as the "signature" of such a matchgate. More precisely, a matchgate is a weighted graph H (which will be used as a local component of a larger graph) with a subset W of its vertices specified as its "external vertices" (which will be used to connect to other matchgates to form a larger graph). Suppose that $(H_1, W_1), (H_2, W_2), \ldots, (H_m, W_m)$ are a collection of matchgates. We may build a larger graph H by connecting these matchgates via their external vertices where the only restriction is that each external vertex of a matchgate (H_i, W_i) connects to exactly one external vertex of a different matchgate (H_j, W_j). The edges that connect the matchgates will be assigned weight 1. Figure 3.2 (a) shows two kinds of matchgates, which are used to construct a larger weighted graph as shown in Figure 3.2 (c) in such a way.

Now let $\mathcal{E}$ denote the set of edges in the larger graph H that connect (the external vertices of) the matchgates, and let $\mathcal{E}(i)$ denote the subset of the edges in $\mathcal{E}$ incident to the external vertices of the matchgate (H_i, W_i). Associate with each $e \in \mathcal{E}$ a $\{0, 1\}$-valued variable x_e. The signature μ_i of the matchgate (H_i, W_i) is a function of the variable set $x_{\mathcal{E}(i)}$ defined as follows: Every configuration $x_{\mathcal{E}(i)}$ is made to correspond to a subgraph

[3] In graph theory, a perfect matching of a graph is a set of non-adjacent edges such that every vertex of the graph is the endpoint of an edge in the set.

of H_i induced by deleting a subset of its external vertices; more precisely, an external vertex is deleted if and only if it is the endpoint of an edge $e \in \mathcal{E}(i)$ whose $x_e = 1$ in the configuration $x_{\mathcal{E}(i)}$; the PerfMatch of the subgraph induced this way is then defined to be the value $\mu_i(x_{\mathcal{E}(i)})$. Then it is possible to express the PerfMatch of the larger graph H as a sum-of-products form involving the signatures of the matchgates as follows.

$$\pi(H) = \sum_{x_{\mathcal{E}}} \prod_{i=1}^{m} \mu_i(x_{\mathcal{E}(i)}). \tag{3.3}$$

It is easy to verify that the sum-of-products form in (3.3) has an NFG representation, since each variable $x_e, e \in \mathcal{E}$, is involved in precisely two functions (noting that every edge $e \in \mathcal{E}$ connects two matchgates). In this case, the NFG has no dangling edges, and the realized exterior function reduces to a scalar, i.e., the PerfMatch of the larger graph H.

Figure 3.2 shows an example of how to build an NFG that realizes the PerfMatch of a graph using the signatures of its matchgates. In the figure, (a) shows two matchgates, where the signature of each matchgate by itself can be viewed as an NFG vertex in (b). When we use the matchgates in (a) to build the larger graph H in (c), then equality (3.3) suggests that the PerfMatch of H is realized by the NFG in (d). That is, the NFG topology is identical to the topology by which the matchgates form the larger graph H. Visually, the relationship between the NFG and the graph H is apparent: The picture in Figure 3.2(d) is the NFG if we ignore the details inside the boxes, and is the graph H if we ignore the boxes.

Solving Counting Problems via Holographic Reduction Many counting problems are described in terms of a large collection of variables and a large collection of constraints each involving a subset of the variables. The objective of such problems is to compute the total number of global variable configurations satisfying all the constraints. In this context, the idea of holographic reduction is to transform the problem of interest to a PerfMatch problem. We now outline this approach.

1. Express the problem as the computation of the exterior function realized by a *planar* NFG $\mathcal{G}$ without dangling edges. When this is possible, each vertex of $\mathcal{G}$

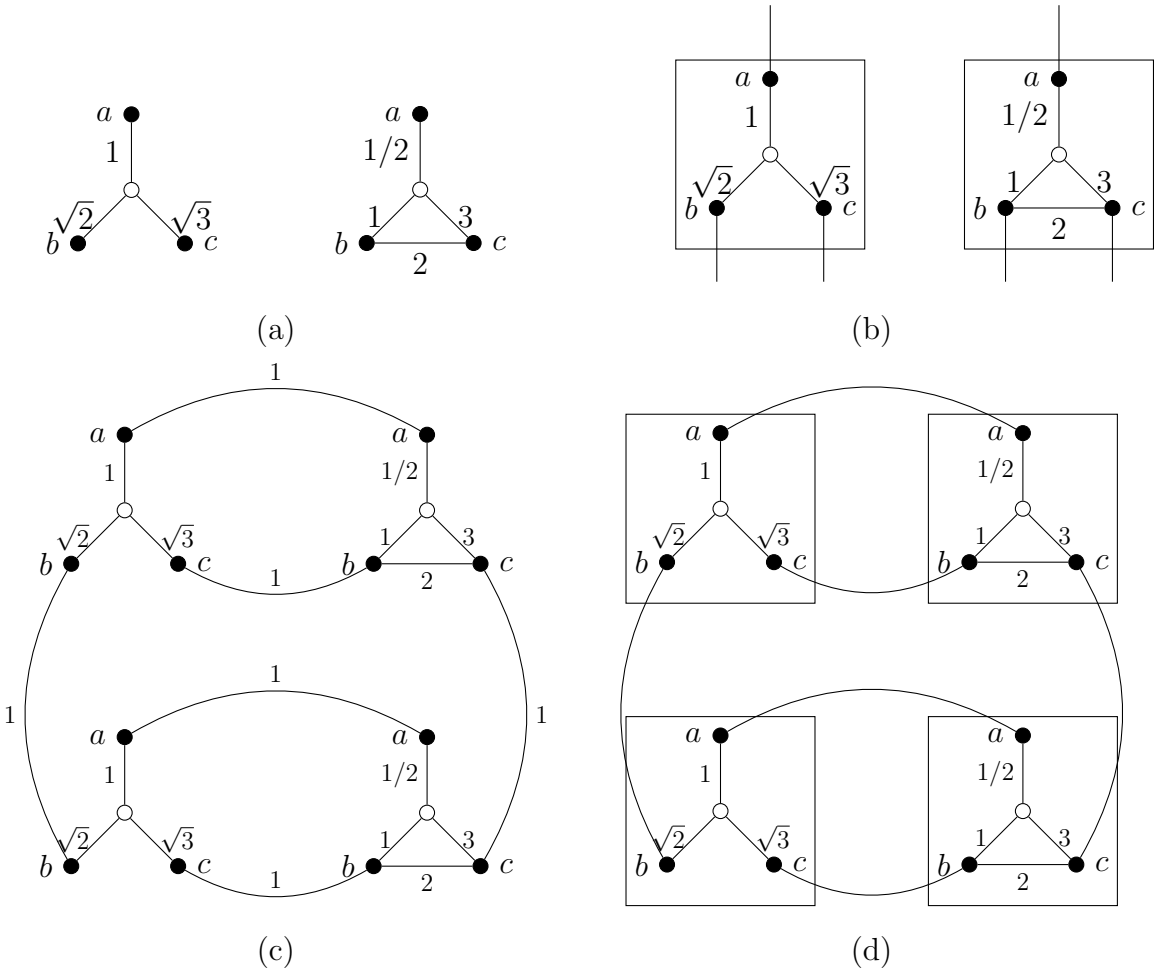


Figure 3.2: (a) Two matchgates, where solid circles represent the external vertices; (b) the signatures of the matchgates represented as NFG function vertices (the boxes); (c) a larger graph H constructed from the matchgates; (d) the NFG realization of the Perf-Match of H, where each box is a function vertex.

represents an indicator (*i.e.*, $\{0,1\}$-valued) function.

2. For each variable x_e in $\mathcal{G}$, find a pair of inverse transformations Φ_e and $\widehat{\Phi}_e$, and construct a holographic transformation $\mathcal{G}^H$ of $\mathcal{G}$ such that each function vertex in $\mathcal{G}^H$ represents the signature of a matchgate.

3. Create a weighted graph H by replacing each vertex of $\mathcal{G}^H$ with the corresponding matchgate drawing, and assign weight one to each edge of $\mathcal{G}^H$. This process essentially turns the holographically transformed NFG as in Figure 3.2(d) into a weighted graph as in Figure 3.2(c).

By the Holant theorem, the exterior function realized by $\mathcal{G}$ is the same as that realized by $\mathcal{G}^H$. Since $\mathcal{G}$ and $\mathcal{G}^H$ do not have dangling edges, the realized exterior function is in fact a scalar; by (3.3), this scalar is the PerfMatch of H. It is easy to verify that $\mathcal{G}$ being a planar graph implies that H is a planar graph (provided that each matchgate is also a planar graph). Thus solving the PerfMatch problem for H via the FKT algorithm solves the original counting problem in polynomial time.

In [61], finding the "right" transformations $\{\Phi_e : e \in E\}$ that transform each local function in the original NFG to the signature of some matchgate appeared to be an art. Later Cai and Lu [9] presented a systematic approach to determine whether such a transformation exists. Remarkably, Cai *et al.* [10] have extended the approach of holographic reduction beyond transformations to the PerfMatch problem by introducing the concept of "Fibonacci gates."

Chapter 4

Normal Factor Graphs and Linear Algebra

In this chapter, we point out yet another direction that NFGs may demonstrate their advantages, namely, their use as a linear algebraic tool. Much of this work is motivated by the notion of "trace diagrams," which has been recently recognized in the mathematics community as an elegant and intuitive "diagrammatic approach" towards expressing notions in linear algebra [19, 48, 53, 54, 56]. Many of the examples in this chapter have trace diagrams counter-parts [48]. Our main objective is to demonstrate how NFGs can be used to provide diagrammatic proofs of some results. We argue that NFGs generalize trace diagrams and so, they may potentially provide a wider range of algebraic tools.

4.1 Notation and conventions

For any positive integer n, let S_n be the symmetric group on $\{1, \ldots, n\}$. A permutation $\sigma \in S_n$ such that $\sigma(j) = i_j$ for all $j \in \{1, \ldots, n\}$ will be written as $\sigma = \begin{pmatrix} 1 & 2 & \cdots & n \\ i_1 & i_2 & \cdots & i_n \end{pmatrix}$. A permutation that exchanges two elements and leaves all other elements unchanged is called a *swap* or a *transposition*. It is well known that any (non-trivial) permutation can be written as a composition of (not necessarily disjoint) swaps. Although the expression of a permutation in terms of swaps is not unique, the parity

(even or odd) of the number of such swaps is unique. If a permutation is the composition of an even number of swaps, then it is said to be *even*, otherwise it is said to be *odd*.

In this chapter, we use ε to denote the Levi-Civita symbol, which is defined as

$$\varepsilon(x_1,\ldots,x_n) = \begin{cases} \operatorname{sgn}\begin{pmatrix} 1 & \cdots & n \\ x_1 & \cdots & x_n \end{pmatrix}, & \begin{pmatrix} 1 & \cdots & n \\ x_1 & \cdots & x_n \end{pmatrix} \in S_n \\ 0, & \text{otherwise} \end{cases}$$

for all $(x_1,\ldots,x_n) \in \{1,\ldots,n\}^n$, where $\operatorname{sgn}(\sigma)$ is the *signature* of the permutation σ defined as $\operatorname{sgn}(\sigma) = 1$ if σ is even and $\operatorname{sgn}(\sigma) = -1$ if σ is odd. Note that the domain of ε is specified by the number of its arguments, and hence we need not be explicit about it. Throughout this chapter, we will liberally make use of the following lemma about the Levi-Civita symbol, which easily follows from its definition.

Lemma 4 *For any positive integer n, it holds that*

$$\varepsilon(x_1,x_2,\ldots,x_{n-1},x_n) = (-1)^{n-1}\varepsilon(x_2,x_3,\ldots,x_n,x_1).$$

In particular, if n is odd, then ε is invariant under cyclic-shifts of its arguments.

In some subsequent figures we may equate an NFG $\mathcal{G}$ to a function f, which strictly speaking is not correct, but what we really mean is that $Z_{\mathcal{G}} = f$. Such practice is unlikely to raise confusion and makes some of the analysis more transparent.

In this chapter we find it helpful to adopt the notion of "ciliation" [48] to explicitly indicate the local orderings of variables at each node. To this end, we add a "dot"on each node to mark the edge carrying the local function's first argument, and we assume the rest of its arguments are encountered in a counter-clockwise manner, cf. Fig. 4.1. We will not insist on ciliations in every occasion and use them only to facilitate the analysis. As an example of the effect of different ciliation arrangements on the realized exterior function, consider the NFGs in Fig. 4.2 where A and B are $n \times n$ matrices. For $i = 1,\ldots,4$, viewing $Z_{\mathcal{G}_i}$ as a matrix with rows and columns indexed by x_1 and x_2, respectively, it is easy to verify that $Z_{\mathcal{G}_1} = AB$, $Z_{\mathcal{G}_2} = AB^T$, $Z_{\mathcal{G}_3} = A^TB^T$, and $Z_{\mathcal{G}_4} = A^TB$, where the product is the regular matrix product.

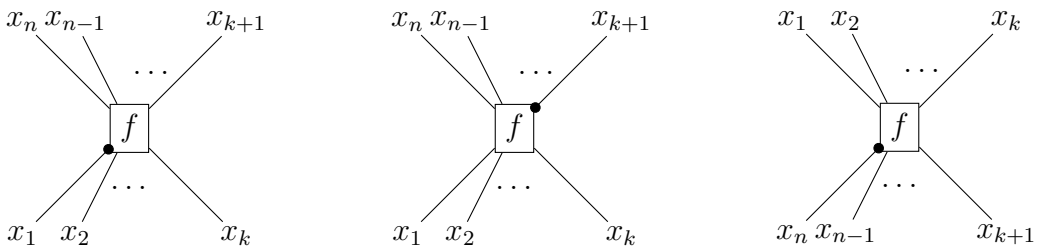


Figure 4.1: Three assignments of ciliations illustrating different orderings of arguments: (a) $f(x_1, \ldots, x_n)$, (b) $f(x_{k+1}, \ldots, x_n, x_1, \ldots, x_k)$ and (c) $f(x_n, x_{n-1}, \ldots, x_1)$.

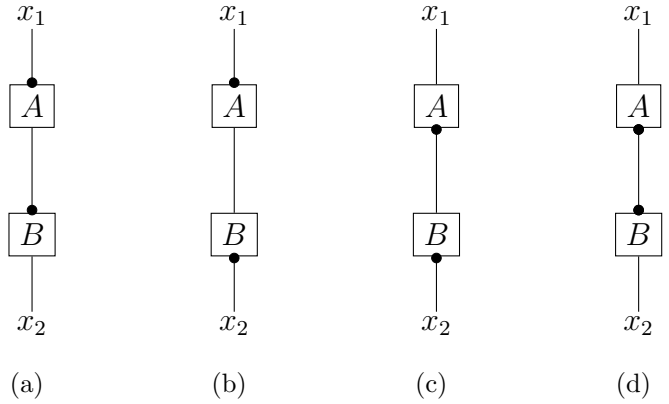


Figure 4.2: Different arrangements of ciliations: (a) $\mathcal{G}_1$, (b) $\mathcal{G}_2$, (c) $\mathcal{G}_3$, and (d) $\mathcal{G}_4$.

We end this section with an example. Consider the NFG $\mathcal{G}$ in Fig. 4.3 (a) where A is an $n \times n$ matrix. Then from the definition of the exterior function, we have

$$Z_{\mathcal{G}} = \sum_{i=1}^{n} A(i,i) = \mathrm{tr}(A).$$

This is the reason behind the name "trace diagrams" in [19, 48, 53, 54, 56]. Briefly, a *trace diagram* is an NFG where each node has degree one, degree two, or is associated a Levi-Civita symbol. As another example, let A be an $m \times n$ matrix and B be an $n \times m$ matrix. Traversing the NFG in Fig. 4.3 (b) in a counter-clockwise way starting from the upper edge, then traversing it again starting from the lower edge illustrates the well-known identity, $\mathrm{tr}(AB) = \mathrm{tr}(BA)$.

(a) (b)

Figure 4.3: (a) An NFG that realizes $\mathrm{tr}(A)$ and (b) an NFG that illustrates $\mathrm{tr}(AB) = \mathrm{tr}(BA)$.

4.2 Derived NFGs: Scaling, addition, and subtraction of NFGs

We have already seen vertex grouping and splitting as a means of manipulating NFGs. Given two NFGs $\mathcal{G}_1$ and $\mathcal{G}_2$ with disjoint dangling edge sets, one of the most basic ways to obtain a new NFG, say $\mathcal{G}$, from $\mathcal{G}_1$ and $\mathcal{G}_2$ is to stack $\mathcal{G}_1$ and $\mathcal{G}_2$ beside each other. Grouping the vertices of $\mathcal{G}_1$ and $\mathcal{G}_2$, it is clear that $Z_{\mathcal{G}}$ is the tensor product of $Z_{\mathcal{G}_1}$ and $Z_{\mathcal{G}_2}$. If $\mathcal{G}_2$ is a single node of degree zero, then $Z_{\mathcal{G}_2}$ is a constant, say λ, and so $Z_{\mathcal{G}} = \lambda Z_{\mathcal{G}_1}$. In this case, we write $\mathcal{G} = \lambda\mathcal{G}_1$ and say $\mathcal{G}$ is a *scaled* version of $\mathcal{G}_1$ with λ being the *scaling* factor. Graphically, we show scaling by putting the scaling factor beside the NFG.[1]

Given two NFGs $\mathcal{G}_1$ and $\mathcal{G}_2$ with the same set of dangling edges, we may graphically construct a *compound* NFG, denoted $\mathcal{G}_1 + \mathcal{G}_2$, by drawing the two graphs with the plus sign "+" between them, see Fig. 4.4. We use this compound NFG to represent the function $Z_{\mathcal{G}_1} + Z_{\mathcal{G}_2}$, and more formally, we say the compound NFG realizes $Z_{\mathcal{G}_1} + Z_{\mathcal{G}_2}$. Finally, the "subtraction of $\mathcal{G}_2$ from $\mathcal{G}_1$" compound NFG, denoted $\mathcal{G}_1 - \mathcal{G}_2$, is defined as $\mathcal{G}_1 + (-1)\mathcal{G}_2$.

The main purpose of the next few sections is to demonstrate how NFGs can be used to establish various identities from linear algebra in a simple and intuitive diagrammatic manner.

[1] An equivalent way to obtain $\mathcal{G}$ directly from $\mathcal{G}_1$ is to replace any function node, say f, in $\mathcal{G}_1$ with the function node λf in $\mathcal{G}$, and to leave everything else the same.

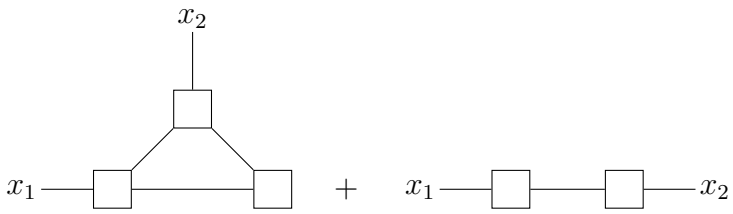


Figure 4.4: Addition of two NFGs.

4.3 Cross product

In this section we look at the cross product of vectors of length 3. Let $\mathcal{G}$ be as in Fig. 4.5, then direct computation shows that $Z_G(1) = u(2)v(3) - u(3)v(2)$, $Z_{\mathcal{G}}(2) = u(3)v(1) - u(1)v(3)$, and $Z_{\mathcal{G}}(3) = u(1)v(2) - u(2)v(1)$. That is, when viewed as vectors in $\mathbb{F}^3$, the exterior function $Z_{\mathcal{G}}$ is the cross product $u \times v$.

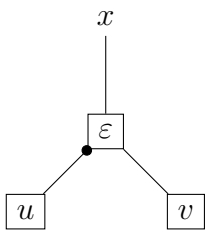


Figure 4.5: Cross product.

A well-known fact, and straightforward to prove, is the following identity about the contraction of two Levi-Civita symbols, namely

$$\sum_t \varepsilon(y_1, y_2, t)\varepsilon(t, x_2, x_1) = \delta(x_1, y_2)\delta(x_2, y_1) - \delta(x_1, y_1)\delta(x_2, y_2).$$

A graphical illustration of the Levi-Civita contraction identity is shown in Fig. 4.6.

Gluing single variate nodes w, s, u, and v, respectively, on the dangling edges x_1, x_2, y_1, and y_2 in Fig. 4.6 results in Fig. 4.7. From the previous analysis (in particular, Lemmas 2 and 4), the following identities are then clear:

$$\begin{aligned}(u \times v) \cdot (s \times w) &= ((u \times v) \times s) \cdot w = (w \times (u \times v)) \cdot s \\ &= ((s \times w) \times u) \cdot v = (v \times (s \times w)) \cdot u \\ &= (u \cdot s)(v \cdot w) - (u \cdot w)(v \cdot s).\end{aligned}$$

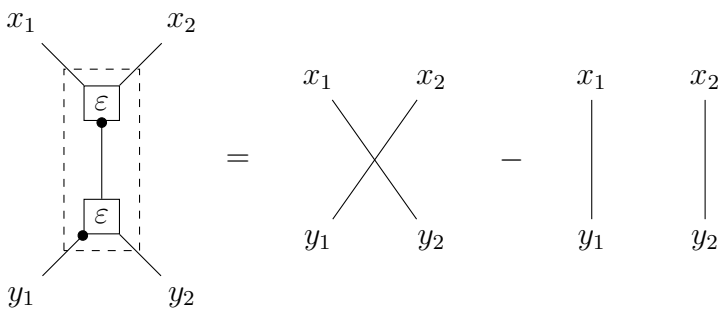


Figure 4.6: Contraction of two Levi-Civita symbols.

Many identities regarding the cross product may be obtained in the same way. For instance, let m_a, m_b, m_c, and m_d be arbitrary positive integers and let $\{a_i : 1 \le i \le m_a\}$, $\{b_i : 1 \le i \le m_b\}$, $\{c_i : 1 \le i \le m_c\}$, and $\{d_i : 1 \le i \le m_d\}$ be four collections of length-three vectors. Further, let A be the matrix whose ith column is a_i, i.e., $A = (a_i : 1 \le i \le m_a)$, and similarly let $B = (b_i : 1 \le i \le m_b)$, $C = (c_i : 1 \le i \le m_c)$, and $D = (d_i : 1 \le i \le m_d)$. We claim that if $m_a = m_d = m$ and $m_b = m_c = m'$ for some m and m', then

$$\sum_{i=1}^{m}\sum_{j=1}^{m'}(a_i \times b_j)\cdot(c_j \times d_i) = \mathrm{tr}(AD^TBC^T) - \mathrm{tr}(BC^T)\mathrm{tr}(AD^T). \tag{4.1}$$

This identity follows from Fig. 4.8 by noting that its left and right hand sides are realized by the left and right NFGs in Fig. 4.8, respectively.

Slight variations of the left NFG in Fig. 4.8 may be used to prove more identities about the cross product. For instance, when $m_a = m_b = m$ and $m_c = m_d = m'$ for some m and m', Fig. 4.9 (a) shows that

$$\sum_{i=1}^{m}\sum_{j=1}^{m'}(a_i \times b_i)\cdot(c_j \times d_j) = \mathrm{tr}(AB^TDC^T) - \mathrm{tr}(AB^TCD^T), \tag{4.2}$$

and when $m_a = 1$ and $m_b = m_c = m$ for some m, Fig. 4.9 (b) proves

$$\sum_{i=1}^{m}(a \times b_i) \times c_i = \left(BC^T\right)\cdot a - \mathrm{tr}\left(BC^T\right)a. \tag{4.3}$$

We invite the reader to verify some of the identities above using traditional methods and contrast with the current diagrammatic approach.

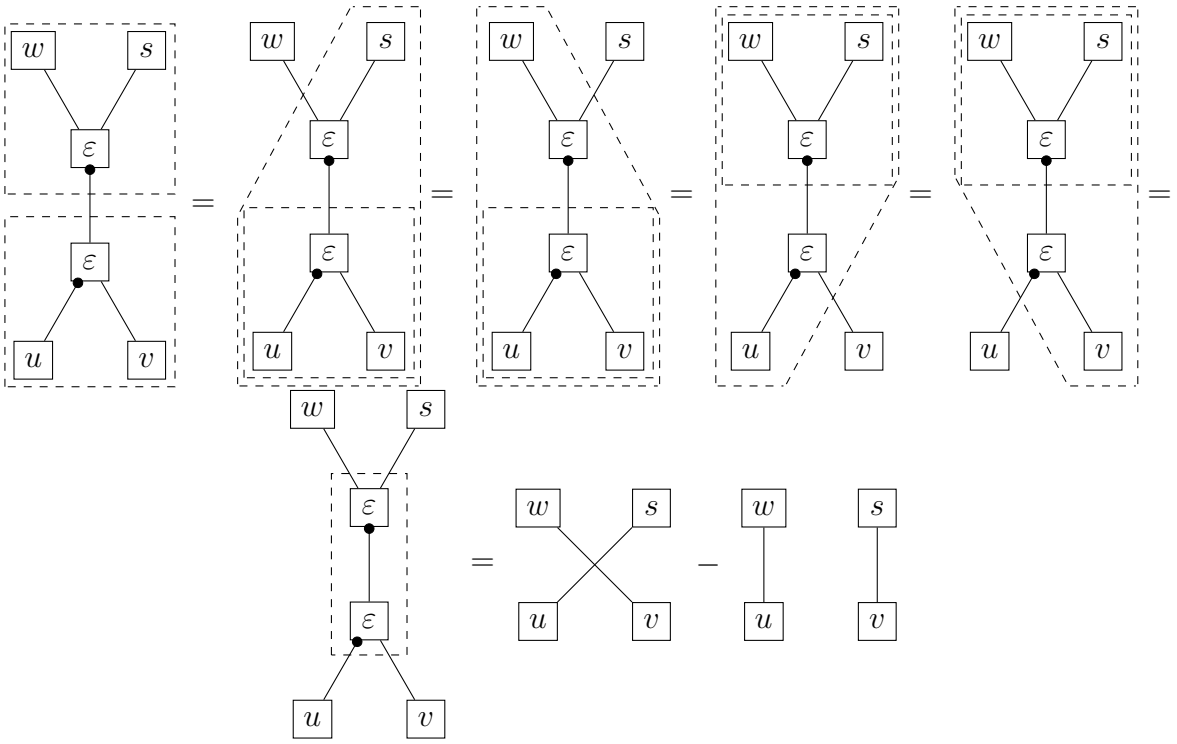


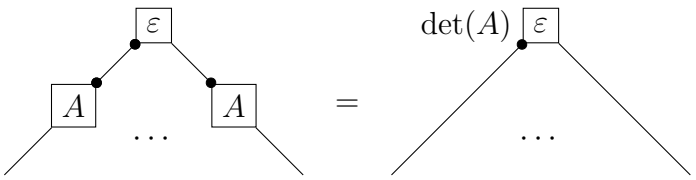
Figure 4.7: A proof of $(u \times v) \cdot (s \times w) = \cdots = (u \cdot s)(v \cdot w) - (u \cdot w)(v \cdot s)$.

4.4 Determinant

For any $n \times n$ matrix A, the determinant, denoted $\det(A)$ or $|A|$, is defined as

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{j=1}^{n} A(j, \sigma(j)).$$

From the definitions of the exterior function and the determinant, it follows that



From this, several properties of the determinant may be obtained, for instance

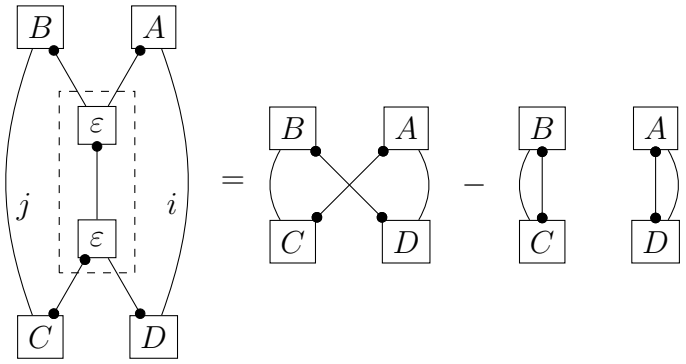


Figure 4.8: A diagrammatic proof of (4.1).

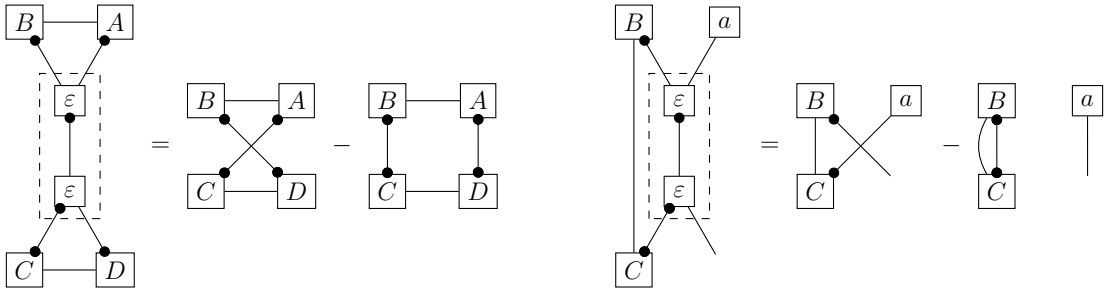
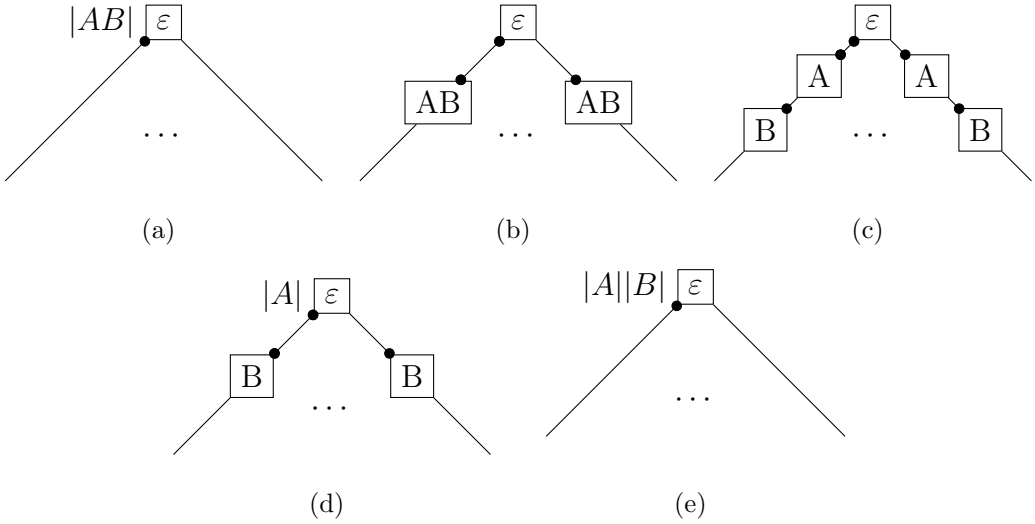
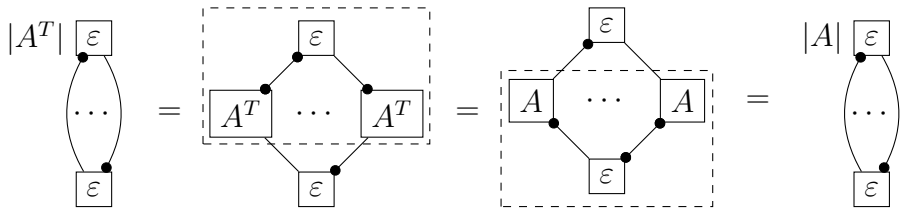


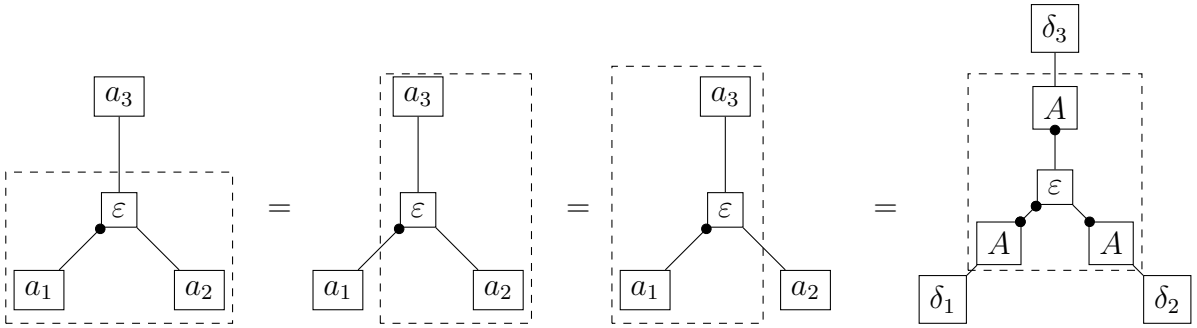
Figure 4.9: Diagrammatic proofs of (4.2) and (4.3).



shows that $\det(AB) = \det(A)\det(B)$ for any $n \times n$ matrices A and B. Another example follows from



showing $\det(A^T) = \det(A)$. As a final example, we have



which proves

$$(a_1 \times a_2) \cdot a_3 = (a_2 \times a_3) \cdot a_1 = (a_3 \times a_1) \cdot a_2 = \det(A),$$

where A is the 3×3 matrix $(a_1\ a_2\ a_3)$ and for each $i \in \{1, 2, 3\}$, $\delta_i(x) = \delta_=(x, i)$ for all $x \in \{1, 2, 3\}$.

4.5 Pfaffian

Given a $2n \times 2n$ skew-symmetric matrix A, the *Pfaffian* of A, denoted $\mathrm{Pf}(A)$, is defined as

$$\mathrm{Pf}(A) = \frac{1}{2^n n!} \sum_{\sigma \in S_{2n}} \mathrm{sgn}(\sigma) \prod_{i=1}^{n} A\big(\sigma(2i-1), \sigma(2i)\big).$$

Before we proceed, we need the following lemma.

Lemma 5 *For any positive integer n, let $\tau \in S_{2n}$ be such that $\tau(2k-1) = k$ and $\tau(2k) = 2n - (k-1)$, $1 \le k \le n$. Then, τ is an even permutation.*

Proof: For brevity, we use the tuple $(i_1, \ldots, i_{2n})$ to refer to the permutation $\begin{pmatrix} 1 & \cdots & 2n \\ i_1 & \cdots & i_{2n} \end{pmatrix}$. Now, given the identity permutation $(1, \ldots, n, n+1, \ldots, 2n)$, it is possible to reflect the second half of the permutation so that it becomes $(1, \ldots, n, 2n, \ldots, n+1)$; this clearly can be done using $\lfloor \frac{n}{2} \rfloor$ swaps. Next, we interleave the two halves of the $2n$-tuple by performing the following $n-1$ steps: Starting with $k = 1$ and ending with $k = n-1$, at step k imagine a window that starts just before the $(2k)$th element of the tuple and ends just after the $(n+k)$th element (i.e., it covers $n-k+1$ elements). For the elements of the tuple covered by the window, we perform a cyclic-shift by one position to the right. It is easy to check that after the $(n-1)$th step we arrive to our desired permutation. Clearly, a one-position cyclic shift on m elements can be accomplished using $m-1$ swaps. Hence, interleaving the tuple requires $\sum_{m=2}^{n}(m-1) = \frac{n(n-1)}{2}$ swaps. Therefore, in total our permutation can be written in terms of $\lfloor \frac{n}{2} \rfloor + \frac{n(n-1)}{2}$ swaps and the claim follows by noting that this number is even for any positive integer n.[2] ■

Below is an example illustrating the lemma and its proof.

Example 1 *Consider for instance* $n = 3$, *then* $\tau = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 6 & 2 & 5 & 3 & 4 \end{pmatrix}$, *and the proof performs the following steps: (Each arrow is labeled with the number of swaps needed.)*

- *Reflection. (Performed using* $\lfloor \frac{3}{2} \rfloor = 1$ *swap.)*

$$1\,2\,3\,4\,5\,6 \xrightarrow{1} 1\,2\,3\,6\,5\,4$$

- *Interleaving. (Performed using* $\frac{3\times 2}{2} = 3$ *swaps.)*

$$1\,\overline{2\,3\,6}\,5\,4 \xrightarrow{2} 1\,6\,2\,\overline{3\,5}\,4 \xrightarrow{1} 1\,6\,2\,5\,3\,4$$

[2]For any positive integer n,

$$\left\lfloor \frac{n}{2} \right\rfloor + \frac{n(n-1)}{2} = \begin{cases} k + k(2k-1), & n = 2k \\ k + (2k+1)k, & n = 2k+1 \end{cases} = \begin{cases} 2k^2, & n = 2k \\ 2k(k+1), & n = 2k+1 \end{cases}.$$

For $n = 4$*, we have* $\tau = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 8 & 2 & 7 & 3 & 6 & 4 & 5 \end{pmatrix}$*, and the proof performs the following steps:*

- *Reflection. (Performed using* $\lfloor \frac{4}{2} \rfloor = 2$ *swaps.)*

$$1\,2\,3\,4\,5\,6\,7\,8 \xrightarrow{2} 1\,2\,3\,4\,8\,7\,6\,5$$

- *Interleaving. (Performed using* $\frac{4\times 3}{2} = 6$ *swaps.)*

$$1\,\overline{2\,3\,4\,8}\,7\,6\,5 \xrightarrow{3} 1\,8\,2\,\overline{3\,4\,7}\,6\,5 \xrightarrow{2} 1\,8\,2\,7\,3\,\overline{4\,6}\,5 \xrightarrow{1} 1\,8\,2\,7\,3\,6\,4\,5$$

For the interested reader, below are the expressions of τ *using the cycle notation from algebra. For* $n = 3$*,* $\tau = (2\,6\,4\,5\,3)$*, from which it is clear that* τ *is even since it has no cycles of even length in its cycle decomposition.*[3] *The proof expresses* τ *in terms of swaps as*

$$\tau = (4\,6)(2\,4\,3)(4\,5) = (4\,6)(2\,3)(2\,4)(4\,5).$$

For $n = 4$*, we have* $\tau = (2\,8\,5\,3)(4\,7)$ *which is clearly even. The proof expresses* τ *in terms of swaps as*

$$\tau = (5\,8)(6\,7)(2\,5\,4\,3)(4\,6\,5)(6\,7) = (5\,8)(6\,7)(2\,3)(2\,4)(2\,5)(4\,5)(4\,6)(6\,7)$$

We remark that the expression of a permutation in terms of non disjoint swaps is not unique. However, the parity (even or odd) of the number of such swaps is unique, which defines the parity of the permutation.

The following proposition affirmatively proves Peterson's conjecture on the Pfaffian [55, 56].

[3] The cycle decomposition expresses a permutation in terms of disjoint cycles, where a *cycle* is a tuple of distinct integers $(i_1 \cdots i_m)$ such that $\sigma(i_j) = i_{j+1}$ for $1 \leq j < m$ and $\sigma(i_m) = i_1$. Cycles of length one are not explicitly written, and it is possible to show that a permutation is odd if and only if the number of cycles of even length in its cycle decomposition is odd.

Proposition 1 *Let A be a $2n \times 2n$ skew-symmetric matrix and let $\mathcal{G}$ be as in Fig. 4.10. Then,*

$$Z_{\mathcal{G}} = n!2^n \mathrm{Pf(A)}.$$

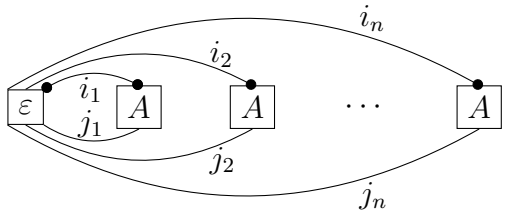


Figure 4.10: The NFG $\mathcal{G}$ from Proposition 1.

Proof: Let $N = \{1, \ldots, n\}$, we have

$$\begin{aligned} Z_{\mathcal{G}} &= \sum_{(i_1,\ldots,i_n,j_n,\ldots,j_1)\in N^{2n}} \varepsilon(i_1,\ldots,i_n,j_n,\ldots,j_1) \prod_{k=1}^{n} A(i_k,j_k) \\ &= \sum_{\sigma\in S_{2n}} \mathrm{sgn}(\sigma) \prod_{k=1}^{n} A\big(\sigma(k), \sigma\left(2n-(k-1)\right)\big). \end{aligned}$$

Now for any $\sigma = \begin{pmatrix} 1 & 2 & \ldots & n & n+1 & \ldots & 2n \\ i_1 & i_2 & \ldots & i_n & j_n & \ldots & j_1 \end{pmatrix} \in S_{2n}$ let $\sigma' = \sigma \circ \tau$, where τ is as in Lemma 5. Then, $\mathrm{sgn}(\sigma') = \mathrm{sgn}(\sigma) \cdot \mathrm{sgn}(\tau) = \mathrm{sgn}(\sigma)$, where the last equality follows from Lemma 5. Further, note that for all $k \in \{1, \ldots, n\}$ we have $\sigma(k) = i_k = \sigma'(2k-1)$ and $\sigma(2n-(k-1)) = j_k = \sigma'(2k)$. Finally, it is clear that as σ runs over S_{2n}, σ' runs over S_{2n}. Hence,

$$Z_{\mathcal{G}} = \sum_{\sigma'\in S_{2n}} \mathrm{sgn}(\sigma') \prod_{k=1}^{n} A\big(\sigma'(2k-1), \sigma'(2k)\big),$$

and the claim follows by the definition of the Pfaffian. ■

Chapter 5

Probabilistic Models

As a notational convention that will be used throughout this chapter, a random variable (RV) is denoted by a capitalized letter, for example, by X, $Y, \ldots$, and the value it takes will be denoted by the corresponding lower-cased letter, i.e., $x, y, \ldots$.

5.1 Probabilistic graphical models

Here we give a brief summary of the previous graphical models relevant to this chapter and develop some notations and definitions for subsequent discussions.

5.1.1 Factor graphs

A *factor graph* (FG) [38] is a bipartite graph $(V \cup U, E)$ with independent vertex sets V and U, and edge set E, where each vertex $v \in V$ is associated a variable x_v from a finite alphabet $\mathcal{X}_v$, and each vertex $u \in U$ is associated a complex-valued function f_u on the Cartesian product $\mathcal{X}_{\mathrm{ne}(u)} := \prod_{v \in \mathrm{ne}(u)} \mathcal{X}_v$, where $\mathrm{ne}(u) := \{v \in V : \{u, v\} \in E\}$ is the set of neighbors (adjacent vertices) of u. Each function f_u is referred to as a *local* function and the FG is said to *represent* a function given by $f(x_V) := \prod_{u \in U} f_u(x_{\mathrm{ne}(u)})$, where we use the "variable set" notation, defined for any $A \subseteq V$, as $x_A := \{x_a : a \in A\}$. In the context of FGs, the function represented by the FG is often called the *global function*. Fig. 5.1 (a) is an example FG.

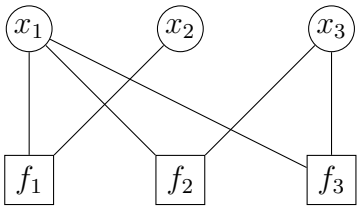


Figure 5.1: An example of a FG, a CFG, and a CDN: (a) When viewed as a FG, the graph represents the global function $f_1(x_1, x_2)f_2(x_1, x_3)f_3(x_1, x_3)$, (b) as a CFG, the graph represents the global function $f_1(x_1, x_2) * f_2(x_1, x_3) * f_3(x_1, x_3)$, and (c) as a CDN, the graph is understood as a FG where each local function is a cumulative distribution, in which case, the global function $f_1(x_1, x_2)f_2(x_1, x_3)f_3(x_1, x_3)$ satisfies the properties of a cumulative function, and is taken as the joint cumulative distribution of the RVs X_1, X_2 and X_3.

Since independence (or conditional independence) relationships among RVs are often captured via the multiplicative factorization of their joint probability distribution, FGs, when used to represent the joint distribution of RVs, form a convenient probabilistic model.

The relationship between FG probabilistic model and other classical probabilistic models, such as Bayesian networks and Markov random fields, is well-known, see, e.g. [38]. In these models, all featuring the "multiplicative semantics" and aiming at representing the joint distributions, efficient inference algorithms, such as the belief propagation or the sum-product algorithm, have been developed and demonstrated great power in various applications.

5.1.2 Convolutional factor graphs

Let $\mathcal{X}_1, \mathcal{X}_2$ and $\mathcal{X}_3$ be three (possibly distinct) finite sets. In general, we require the sets to have an abelian group structure, so that a notion of addition "+" and its inverse "−" are well defined. The requirement that the sets be finite is not particularly critical but only for the convenience of argument.

Let f_1 and f_2 be two function on $\mathcal{X}_1 \times \mathcal{X}_2$ and $\mathcal{X}_2 \times \mathcal{X}_3$, respectively. The *convolution*

of f_1 and f_2, denoted $f_1 * f_2$, is defined as the function on $\mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{X}_3$ given by, $(f_1 * f_2)(x_1, x_2, x_3) := \sum_{x \in \mathcal{X}_2} f_1(x_1, x_2 - x) f_2(x, x_3)$. Following the convention in [45], we may write $(f_1 * f_2)(x_1, x_2, x_3)$ as $f_1(x_1, x_2) * f_2(x_2, x_3)$ to emphasize the domains of the original functions. It is not hard to show that the convolution as defined above is both associative and commutative.

A *convolutional factor graph* (CFG) [44] is a bipartite graph that represents a global function that factors as the convolution of local functions. In fact, the representation semantics in a CFG is identical to that in a FG (often referred to in this chapter as a "multiplicative" FG for distinction), except that the above defined notion of convolution is used as the product operation, cf. Fig. 5.1 (b) for an example CFG. In [45] CFGs were presented as a probabilistic graphical model to represent the joint probability distribution of a set of observed RVs that are constructed from a collection of independent sets of unobserved or "latent " RVs via linear combinations. In addition, the authors of [44] presented an elegant duality result between FGs and CFGs via the Fourier transform. Such a duality and CFGs have recently been exploited by [6] in what is known as linear characteristic model (LCM) for solving inference problems with stable distributions.

5.1.3 Cumulative distribution networks

Let X be a RV assuming its values from a finite ordered set $\mathcal{X}$. The *cumulative distribution function* (CDF) of X is defined as $F_X(x) := \sum_{y \leq x} p_X(y)$, where p_X is the probability distribution of X. We note that this definition of CDF, as a function on $\mathcal{X}$, is slightly different from the classical definition of CDF, which is a function defined on the real line (or on the Euclidean space in the multivariate case). It nevertheless captures the same essence and is merely a different representation, suitable and convenient in the context of this work. Such a notion of CDF can be extended to any collection of RVs $X_1, \ldots, X_n$ assuming their values from the finite ordered sets $\mathcal{X}_1, \ldots, \mathcal{X}_n$ by defining their joint CDF as $F_{X_1,\ldots,X_n}(x_1, \ldots, x_n) := \sum_{y_1 \leq x_1, \ldots, y_n \leq x_n} p_{X_1,\ldots,X_n}(y_1, \ldots, y_n)$, where $p_{X_1,\ldots,X_n}$ is the joint probability distribution of $X_1, \ldots, X_n$. Note that while the marginal probability distribution is computed by *summing* the joint probability distribution over the range of the marginalized RVs, the marginal CDF is computed by *evaluating* the joint

CDF at the largest element of $\mathcal{X}_i$, for all marginalized RVs X_i. (That is, if I indexes the set of marginalized RVs and X_i takes its values from the ordered set $\{1, \ldots, |\mathcal{X}_i|\}$, then we evaluate the joint CDF at $|\mathcal{X}_i|$ for all $i \in I$.) It is well known that CDFs satisfy a collection of properties as were articulated in standard textbooks and in [30]. On the other hand, any function satisfying such properties, which we shall refer to as "CDF axioms," may be regarded as a CDF and can be used to define a collection of RVs.

A *cumulative distribution network* (CDN) [30] is a multiplicative FG in which each local function satisfies the CDF axioms, then it is straightforward to show that the global function represented by the FG also satisfies the CDF axioms. The global function thus defines a collection of random variables, each represented by a variable node in the FG, and the CDN may serve as a probabilistic model. In [30], it was shown that CDNs are useful for structured ranking problems, and efficient inference algorithms for such problems were developed in these models. See Fig. 5.1 (c) for an example CDN.

5.1.4 Normal factor graphs

To facilitate notation, in this chapter we use L and T to denote half edges and internal edges, respectively, and we use E to denote both types of edges. For any vertex v, we use $L(v), T(v)$, and $E(v)$ to denote the corresponding type of edges incident on v. Recall that we say two NFGs are *equivalent* if they realize the same exterior function. At some places, we may extend this notion of equivalence to include other graphical models and say, for instance, "a FG is equivalent to an NFG," where we mean that the product function of the FG is equal to the exterior function of the NFG. We call an NFG with no loops or parallel internal edges a *simple* NFG, and an NFG with a bipartite underlying graph $(I \cup J, E)$ a *bipartite* NFG, where I and J are the two independent vertex sets. We impose no restriction on the cycle structure of NFGs.

The following (non-disjoint) classes of local functions will be of particular interest.

Split functions Let $\mathcal{X}_1, \ldots, \mathcal{X}_n$ be some finite sets, then we say a function f on $\mathcal{X}_1 \times \cdots \times \mathcal{X}_n$ is a *split* function via x_1, and refer to x_1 as the *splitting variable*, if

$$f(x_1, \ldots, x_n) = f_2(x_1, x_2) f_3(x_1, x_3) \ldots f_n(x_1, x_n),$$

for some bivariate functions $f_2, \ldots, f_n$. Note that it follows immediately that any bivariate function is trivially a split function (via any of its arguments). Subsequently, if we do not explicitly specify the splitting variable of a split function, then it is assumed to be the function's first argument. Graphically, we draw a split function as in Fig. 5.2 (a), where the in-ward directed edge is used to distinguish the splitting argument, and the remaining arguments are successively encountered in a counter clock-wise manner with respect to the directed edge.

Conditional functions Let $\mathcal{X}_1, \ldots, \mathcal{X}_n$ be some finite sets, then a function f on $\mathcal{X}_1 \times \cdots \times \mathcal{X}_n$ is said to be a *conditional* function of x_1 given $x_2, \ldots, x_n$ if there is a constant c such that $\sum_{x_1} f(x_1, \ldots, x_n) = c$, for all $x_2, \ldots, x_n$. It is apparent that a non-negative real conditional function with $c = 1$ is a conditional probability distribution. A conditional function is shown in Fig. 5.2 (b), where we use the same convention of edge labeling as in the case of split functions, but with an out-ward directed edge to mark the first argument.

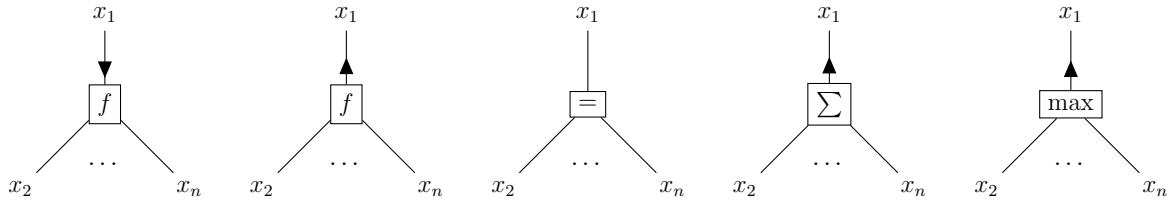


Figure 5.2: A graphical illustration of: (a) split function, (b) conditional function, (c) $\delta_=$, (d) δ_Σ, and (e) $\delta_{\max}$.

One may observe a sense of "duality" between a conditional function and a split function through the following lemma.

Lemma 6 *Let $f(x_1, x_2, x_3)$ be a positive real split function, then up to a scaling factor, f may be written as $p_{X_1}(x_1)p_{X_2|X_1}(x_2|x_1)p_{X_3|X_1}(x_3|x_1)$ for some probability distributions p_{X_1}, $p_{X_2|X_1}$, and $p_{X_3|X_1}$.*

Proof: Since f is a positive real function, it may be viewed (up to a scaling factor) as a probability distribution of some RVs X_1, X_2 and X_3. Hence, f can be written as

$f(x_1, x_2, x_3) = p_{X_1}(x_1)p_{X_2|X_1}(x_2|x_1)p_{X_3|X_1X_2}(x_3|x_1, x_2)$. Since f is a split function via x_1, as we will see later (cf. Lemma 9) , we have $X_2 \perp\!\!\!\perp X_3|X_1$, and the claim follows. ■

Now compare a split function $f(x_1, x_2, x_3)$ with a conditional function $g(x_1, x_2, x_3)$ where let us assume that the respective scaling constants making the functions into distributions are both 1. If we are to draw the Bayesian networks (BN) [52] corresponding to the two distributions f and g respectively, we shall see that the directions of the edges in the BN of f are completely opposite to those in the BN of g. Describing it in terms of causality, one may say: The distribution f prescribes that conditioned on the RV X_1, we generate the RVs X_2 and X_3 *independently*, whereas the distribution g prescribes that X_2 and X_3 generate X_1 *jointly*. This sense of "duality" or "reciprocity" (evidently existing in the two kinds of functions involving arbitrary number of variables) also justifies our notations of opposite edge directions in denoting the two kinds of functions.

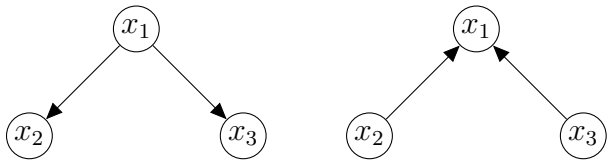


Figure 5.3: The BNs corresponding to: (a) a split function, and (b) a conditional function.

Evaluation indicator Let $\mathcal{X}$ be a finite alphabet, the *evaluation indicator* (evaluating at some $\overline{x} \in \mathcal{X}$) is an indicator function on $\mathcal{X}$ defined as $\delta_{\overline{x}}(x) := [x = \overline{x}]$ for all $x \in \mathcal{X}$,

Equality indicator Let $\mathcal{X}$ be a finite alphabet, recall that the *equality indicator* on n variables is defined as $\delta_=(x_1, \ldots, x_n) := \prod_{i=2}^{n} [x_1 = x_i]$ for all $x_1, \ldots, x_n \in \mathcal{X}$. Note that an equality indicator is a split function via any of its arguments, hence, at many places, we may illustrate it graphically without a directed edge.

Constant-one indicator Let $\mathcal{X}$ be a finite alphabet, the *constant-one indicator* is a degenerate indicator function defined as $\mathbf{1}(x) := 1$ for all $x \in \mathcal{X}$.

Sum indicator Let $\mathcal{X}$ be a finite abelian group (additively written), the *sum indicator* on n variables is defined as $\delta_\Sigma(x_1, \ldots, x_n) := [x_1 = x_2 + \cdots + x_n]$ for all $x_1, \ldots, x_n \in \mathcal{X}$. A closely related indicator function, which is more popular in the factor graphs and normal graphs literature, is the *parity indicator* function of Chapter 3, denoted δ_+,

and defined as $\delta_+(x_1, \ldots, x_n) := [x_1 + \cdots + x_n = 0]$. It is clear that $\delta_\Sigma(x_1, \ldots, x_n) = \delta_+(x_1, -x_2, \ldots, -x_n) = \delta_+(-x_1, x_2, \ldots, x_n)$.

An elementary result concerning the sum indicator function is the following lemma.

Lemma 7 *For any functions f on $\mathcal{X}_1 \times \mathcal{X}_2$ and g on $\mathcal{X}_2 \times \mathcal{X}_3$, where $\mathcal{X}_1$, $\mathcal{X}_2$ and $\mathcal{X}_3$ are abelian groups,*

$$\sum_{t,u} f(x,t)g(u,z)\delta_\sum(y,t,u) = f(x,y) * g(y,z),$$

where $\delta_\sum$ above is defined on $\mathcal{X}_2^3$.

Proof: Follows directly from the definition of the convolution. ■

Max indicator Let $\mathcal{X}$ be an ordered finite set. The *max indicator* on n variables is defined as $\delta_{\max}(x_1, \ldots, x_n) := [x_1 = \max(x_2, \ldots, x_n)]$ for all $x_1, \ldots, x_n \in \mathcal{X}$. Let I be a finite set and let $\mathcal{X}_i$ be an ordered finite set for all $i \in I$. The definition of the max indicator is extended to the partially-ordered set $\mathcal{X}_I$ by defining $\delta_{\max}(x_I, \ldots, x'_I) := \prod_{i \in I} \delta_{\max}(x_i, \ldots, x'_i)$ for all $x_I, \ldots, x'_I \in \mathcal{X}_I$.

A graphical illustration of the above local functions is shown in Fig. 5.2. Note that the max and sum indicator functions are both conditional functions as illustrated by the directed edges in Figs. 5.2 (d) and (e). It is worth noting that the bivariate max indicator and bivariate sum indicator are both equivalent to the bivariate equality indicator, which is a split and a conditional function.

5.2 Normal factor graph models

We now present a generic NFG probabilistic model. Formally, an *NFG probabilistic model*, or simply an *NFG model*, is an NFG whose exterior function is up to scale the joint distribution of some RVs (each represented by a half edge) and which satisfies the following two properties: 1) the NFG is bipartite and simple; 2) half edges are only incident on one independent vertex set and there is exactly one half edge incident on each vertex in this set; we call these vertices *interface vertices*, and call the ones in the other vertex set *latent vertices*. We will call the corresponding functions indexed by these two vertex sets *interface functions* and *latent functions* respectively, although we will be

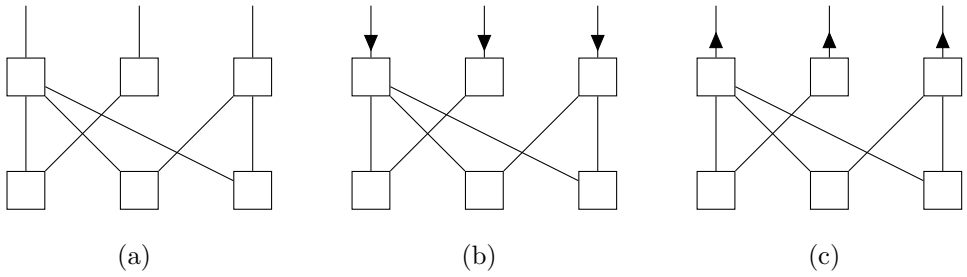


Figure 5.4: (a) An NFG model, (b) a constrained NFG model, and (c) a generative NFG model.

quite loose in speaking of a vertex and a function exchangeably as we do for a variable and an edge/half edge. We will customarily denote the set of interface vertices by I and the set of latent vertices by J.[1]

Note that since each interface function has exactly one half edge incident on it, unless it is more convenient to make the distinction, we will subsequently identify the set of half edges using I, i.e., an interface vertex will index both its function and the half edge incident on it. An example NFG model is shown in Figure 5.4 (a), where the top-layer vertices are the interface vertices, and the bottom-layer vertices are the latent vertices. If necessary, we may formally denote such NFG using such a notation as $\mathcal{G}(I \cup J, E, f_{I \cup J})$.

In this modelling framework, we will focus on two "dual" families of models, which we call the *constrained NFG models* and the *generative NFG models* respectively (see, e.g., Figure 5.4 (b) and (c) respectively for a quick preview). We will demonstrate how these models are related to the previous models such as FG and CFG. We will also introduce "transformed NFG models" in Section 5.3, a special case of which reduces to CDN.

[1]We note that demanding no half edge incident on the latent functions entails no loss of generality, since if there is such a half edge, one may always insert a bivariate equality indicator function (or equivalently a bivariate max indicator or sum indicator) into the half edge, which converts the NFG to an equivalent one with this half edge turned into a regular edge. Since the bivariate equality indicator is both a split function and a conditional function, inserting such a function has no impact on our later restriction on the interface functions, where we require them to be all split functions or all conditional functions.

5.2.1 Constrained NFG model

A *constrained* NFG model is an NFG model in which all interface functions are split functions via their respective external variables.

To bring more intuition into this definition, we first take a slight digression and show in the following lemma that it is possible to "shape" a distribution by "random rejection".

Lemma 8 *Let X be a RV with a probability distribution p_X, where X assumes its values from a finite set $\mathcal{X}$, and let h be a normalized non-negative real function on $\mathcal{X}$ with a non-empty support, where the normalization is in the sense that $\max_{x\in\mathcal{X}} h(x) = 1$. Draw x from p_X and accept it with probability $h(x)$ and reject it with probability $1 - h(x)$. If x is accepted, output x; otherwise repeat the process until some other x' is drawn and accepted. Denote the output random variable by Y. Then the probability distribution p_Y of Y is, up to scale, $h(y)p_X(y)$, for all $y \in \mathcal{X}$.*

Proof: Let Z be a $\{0,1\}$ RV representing the random rejection in the lemma, i.e., a sample $x \in \mathcal{X}$ is rejected if $Z = 0$, accepted if $Z = 1$, and the probability that $Z = 1$ is $h(x)$. Then the statement $Y = y$ is equivalent to $(X, Z) = (y, 1)$, and we have $p_Y(y) = p(X = y)p(Z = 1|X = y) = p_X(y)h(y)$. ■

The idea of "distribution shaping" via "random rejection" is central to the semantics of constrained NFG models, which we demonstrate in the next example.

Example 2 *Let $\mathcal{G}$ be a constrained NFG as in Fig. 5.5 where $f_1(x_1, s_1, s_1')$ and $f_2(x_2, s_2, s_2')$ are positive functions that split via x_1 and x_2, respectively, and h_1, h_2 and h_3 are non-negative functions (with non-empty supports). From Lemma 6 we may express f_1 and f_2, up to a respective scaling factor, as*

$$f_1(x_1, s_1, s_1') = p_{X_1}(x_1)p_{S_1|X_1}(s_1|x_1)p_{S_1'|X_1}(s_1'|x_1)$$

and

$$f_2(x_2, s_2, s_2') = p_{X_2}(x_2)p_{S_2|X_2}(s_2|x_2)p_{S_2'|X_2}(s_2'|x_2),$$

for some distributions p_{X_1}, p_{X_2}, $p_{S_1|X_1}$, $p_{S_1'|X_1}$, $p_{S_2|X_2}$ and $p_{S_2'|X_2}$. The RVs represented by the NFG may be regarded as being generated by the following process.

1. *Draw* (x_1, x_2) *from distribution* $p_{X_1}(x_1)p_{X_2}(x_2)$ *where* p_{X_1} *and* p_{X_2} *are as specified by our choices above. Note that the two components of the drawn vector are independent.*

2. *Draw vector* (s_1, s_1') *from the distribution* $p_{S_1|X_1}(s_1|x_1)p_{S_1'|X_1}(s_1'|x_1)$ *and draw* (s_2, s_2') *from* $p_{S_2|X_2}(s_2|x_2)p_{S_2'|X_2}(s_2'|x_2)$*. It is clear that the joint distribution of the vector* $(x_1, x_2, s_1, s_1', s_2, s_2')$ *is up to scale* $f_1(x_1, s_1, s_1')f_2(x_2, s_2, s_2')$*.*

3. *Let* $H(s_1, s_1', s_2, s_2') := c \cdot h_1(s_1)h_2(s_1', s_2)h_3(s_2')$*, where* c *is a normalizing constant such that the maximum value of* $H(\cdot)$ *is* 1*. Accept the drawn* $(x_1, x_2, s_1, s_1', s_2, s_2')$ *with probability* $H(s_1, s_1', s_2, s_2')$ *and reject it with probability* $1 - H(s_1, s_1', s_2, s_2')$*.*

4. *If the drawn* $(x_1, x_2, s_1, s_1', s_2, s_2')$ *is rejected, repeat the procedure from step 1, until the drawn* $(x_1, x_2, s_1, s_1', s_2, s_2')$ *is accepted. By Lemma 8, the accepted vector has a distribution equal, up to scale, to* $f_1(x_1, s_1, s_1')f_2(x_2, s_2, s_2')H(s_1, s_1', s_2, s_2')$*.*

5. *Output* (x_1, x_2)*. Then clearly the output vector has distribution that is up to scale the exterior function of the NFG.*

The procedure introduced in the example above generalizes in an obvious way to arbitrary constrained NFG models. Instead of precisely, but repetitively, stating the procedure for the general setting, we make the following remarks. The interface functions completely specify how the external variables are drawn and how the internal variables are drawn conditioned on the drawn external configuration. The drawn internal configuration then undergoes a "random rejection" according to the product of all latent functions. The external configuration giving rise to an accepted internal configuration then necessarily follows the distribution prescribed by the exterior function of the NFG.

Analogously, one may view a constrained NFG model as a "probabilistic checking system": Independent "inputs" (external variables) excite the "internal states" (internal variables) of the system via interface functions; the state configuration is "checked" probabilistically by the latent functions; only the external configurations that pass the internal check are kept. In general, the internal checking mechanism induces dependence

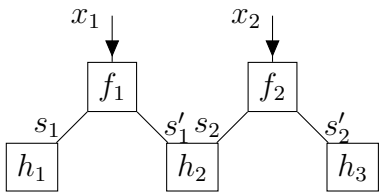


Figure 5.5: Example 2.

among the external variables, which were *a priori* independent. In the special case when the latent functions are all indicator functions, the checking system is in fact deterministic, reducing to a set of constraints on the internal states, cf. Section 5.4. This has been the motivation behind the name "constrained NFG model". As we will show momentarily that constrained NFG models and FG models are equivalent, the "probabilistic checking system" perspective of constrained models provides a different and new interpretation of the FG models.

5.2.2 Constrained NFG models are equivalent to FGs

Suppose that a constrained NFG model is such that every interface function is an equality indicator function. It is known [23] that one may convert such NFG to a FG according to the following procedure: *For each interface vertex, replace it by a variable vertex representing its half-edge variable and remove the half edge.*

Proposition 2 *If in a constrained NFG model all interface functions are equality indicators, then the above procedure gives rise to a FG equivalent to the NFG.*

Proof: Let $\mathcal{G}(I \cup J, E, f_{I\cup J})$ be the NFG in hand where $f_i = \delta_=$ for all $i \in I$. The resulting FG has an underlying graph $(I \cup J, E)$ where I and J are the variable and function indexing sets, respectively. Hence, the global function of the FG is the multiplication $\prod_{j\in J} f_j(x_{\mathrm{ne}(j)})$. On the other hand, if we use $T(v)$ to denote the set of internal edges incident on node v in the NFG, then the exterior function of the NFG is $\left\langle \prod_{j\in J} f_j(s_{T(j)}), \prod_{i\in I} \delta_=(x_i, s_{T(i)}) \right\rangle$, which, if i' and j' are connected by an edge t, accounts to substituting $x_{i'}$ in place of the argument s_t of $f_{j'}$ in the product $\prod_{j\in J} f_j(s_{T(j)})$, for all adjacent i' and j'. The claim follows by noting that $T(j) = \{\{i, j\} : i \in \mathrm{ne}(j)\}$. ■

The proposition essentially suggests that the joint distribution represented by such a constrained NFG model factors multiplicatively and therefore can be represented by a FG. In fact the converse is also true, namely that any FG can be converted to an equivalent constrained NFG model with all interface functions being equality indicators. This is illustrated in Figure 5.6.

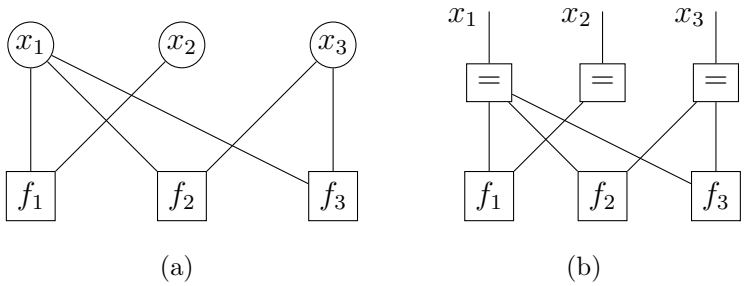


Figure 5.6: A FG and its equivalent constrained NFG.

Next we show that any constrained NFG is in fact equivalent to one with equality interface function. Given an arbitrary constrained NFG $\mathcal{G}$ where each interface function f_i splits as $\prod_{t\in T(i)} f_t$ for some bivariate functions f_t. The following procedure converts $\mathcal{G}$ into a constrained NFG with the same underlying graph as $\mathcal{G}$, and in which all interface functions are equality indicators:

1) Replace each interface function with an equality indicator.

2) Replace each hidden function f_j with $\big\langle f_j, f_t : t \in T(j)\big\rangle$.

Proposition 3 *In the above procedure, the original and resulting constrained NFGs are equivalent, and have the same underlying graph.*

Proof: The fact that the two NFGs have the same underlying graph is clear. To prove equivalence, each interface function f_i is the product $\prod_{t\in T(i)} f_t(x_i, s_t)$ of bivariate functions f_t, and from Proposition 2, it can be written as the sum-of-products form $\big\langle \prod_{t\in T(i)} f_t(s'_t, s_t), \delta_=(x_i, s'_{T(i)})\big\rangle$. By vertex splitting, each interface vertex can be replaced with the NFG representing its corresponding sum-of-product form. The claim follows upon vertex merging each hidden node f_j with its adjacent bivariate functions, i.e., by replacing f_j with $\big\langle f_j, f_t : t \in T(j)\big\rangle$. ■

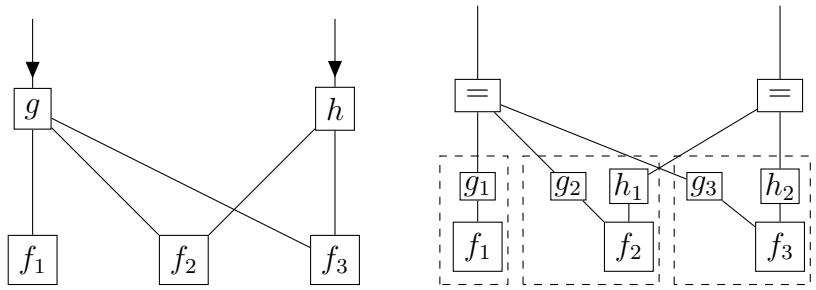


Figure 5.7: Converting a constrained NFG model to one in which all interface functions are equality indicators. (a) An example of a constrained NFG where by assumption g splits into g_1, g_2 and g_3, and h splits into h_1 and h_2, (b) vertex splitting of interface nodes, followed by vertex merging of each hidden function with its neighboring bivariate functions.

The conversion stated in the proposition and an illustration of the proof are shown in Figure 5.7. Invoking Proposition 2, the following theorem is immediate.

Theorem 3 *Every constrained NFG can be converted to an equivalent FG.*

5.2.3 Generative NFG model

A *generative* NFG model is an NFG model in which every interface function is a conditional function of its half edge variable given its remaining arguments. The following example gives sufficient insight of the modelling semantics of a generative NFG.

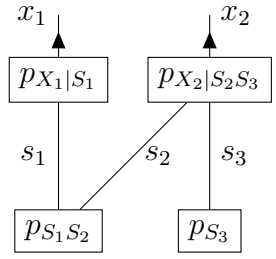


Figure 5.8: Example 3.

Example 3 *Let (S_1, S_2) be jointly distributed according to $p_{S_1S_2}$ and S_3 be distributed according to p_{S_3}, where (S_1, S_2) is independent of S_3. Suppose that (X_1, X_2) depends on (S_1, S_2, S_3) according to the conditional distribution*

$$p_{X_1X_2|S_1S_2S_3}(x_1, x_2|s_1, s_2, s_3) := p_{X_1|S_1}(x_1|s_1)p_{X_2|S_2S_3}(x_2|s_2, s_3),$$

it is then easy to verify that the joint distribution $p_{X_1X_2}(x_1, x_2)$ is given by the sum-of-products form $\langle p_{S_1S_2}, p_{S_3}, p_{X_1|S_1}, p_{X_2|S_2S_3}\rangle$, and hence, the RVs (X_1, X_2) are represented by the generative NFG in Fig. 5.8.

The NFG in this example is a generative NFG model where $p_{X_1|S_1}$ and $p_{X_2|S_2S_3}$ are interface functions and $p_{S_1S_2}$ and p_{S_3} are latent functions. In this case, the latent functions serve as independent sources of randomness, which "generate" the internal RVs (S_1, S_2 and S_3). The internal RVs then "generate" the external RVs via the interface functions.

In an arbitrary generative NFG model, since every latent function may be viewed as the joint distribution of its involved internal RVs, subject to a scaling factor, they can be regarded as independent "generating sources"; since each interface function is a conditional function or, up to a scale, a conditional distribution of the external RV given its internal RVs, the product of these conditional functions may be regarded as the conditional distribution of all external RVs conditioned on the internal RVs. The product of all local functions is then up to scale the joint distribution of all external and internal RVs. The semantics of NFG then dictates that the joint distribution is summed over all the internal variables, and the resulting exterior function is therefore the distribution of the external RVs, up to scale. In a sense, a generative NFG model describes how the external random variables are generated from some independent hidden sources.

5.2.4 A subclass of generative NFG models is equivalent to CFGs

In this section, we rely on the Fourier transform in some of the discussions. Let $\mathcal{X}$ be a finite abelian group. As usual, we use $\mathcal{X}^\wedge$ to denote the character group (written additively) of $\mathcal{X}$, $\kappa_{\mathcal{X}}(x, \hat{x})$ to denote both $x(\hat{x}) = \hat{x}(x)$, and $\hat{\kappa}_{\mathcal{X}}(x, \hat{x})$ to denote

$\kappa(x, -\hat{x})/|\mathcal{X}|$. For any function on $\mathcal{X}$, its Fourier transform is the sum-of-product form $\widehat{f}(\hat{x}) := \langle \kappa_{\mathcal{X}}(x, \hat{x}), f(x) \rangle$ for all $\hat{x} \in \mathcal{X}^{\wedge}$, and one may recover f using the Fourier inverse as $f(x) = \langle \widehat{f}(\hat{x}), \hat{\kappa}_{\mathcal{X}}(x, \hat{x}) \rangle$ for all $x \in \mathcal{X}$. Finally, if $\mathcal{X}$ is the direct product $\prod_{i \in I} \mathcal{X}_i$ of the finite abelian groups $\mathcal{X}_i$, then $(\mathcal{X})^{\wedge}$ is the direct product $\prod_{i \in I} \mathcal{X}_i^{\wedge}$, and it follows that $\kappa_{\mathcal{X}}(x, \hat{x}) = \prod_{i \in I} \kappa_{\mathcal{X}_i}(x_i, \hat{x}_i)$, for all $(x, \hat{x}) \in \mathcal{X} \times \mathcal{X}^{\wedge}$, and similarly for $\hat{\kappa}_{\mathcal{X}}$.

Suppose that a generative NFG model is such that every interface function is a sum indicator function. We may convert such an NFG to a CFG according to the following procedure: *For each interface vertex, replace it by a variable vertex representing its half-edge variable and remove the half edge.*

Proposition 4 *If in a generative NFG model all interface functions are sum indicators, then the above procedure gives rise to a CFG equivalent to the NFG.*

Proof: The proof follows the following steps: 1) Modify the NFG by replacing each interface function with a parity check indicator and inserting a degree two parity check indicator (a sign inverter) on each half edge, Fig. 5.9 (b). (This does not alter the exterior function due to the relation between the sum and the parity indicator function.) 2) Perform a holographic transformation on the resulting NFG by inserting the inverse-pair $\kappa_{\mathcal{X}_e}$ and $\hat{\kappa}_{\mathcal{X}_e}$ into each regular edge e (with $\kappa_{\mathcal{X}_e}$ adjacent to a hidden function or an inserted sign inverter), and inserting the transformers $\kappa_{\mathcal{X}_e}$ into each dangling edge e, Fig. (c). 3) By noting that (up to a scaling factor[2]) the Fourier and Fourier inverse of δ_+ are $\delta_=$, we obtain (after deleting all degree-two equality indicators resulting from the sign inverters) a constrained NFG in which each interface function is an equality indicator and each hidden function is the Fourier transform of the corresponding hidden function in the original NFG, Fig. (d). Hence, from the GHT and Proposition 2, we have (up to a scaling factor) $\widehat{Z}_{\mathcal{G}}(\hat{x}_I) = \prod_{j \in J} \widehat{f}_j(\hat{x}_{\mathrm{ne}(j)})$, and the claim follows from the multivariate multiplication-convolution duality theorem under the Fourier transform [44]. ■

An example illustrating the steps of the proof is shown in Fig. 5.9. Of course one may attempt to prove the claim by direct evaluation of the exterior function as demonstrated

[2]It is not hard to show that all the scaling factors cancel out, and hence, all subsequent equalities are exact.

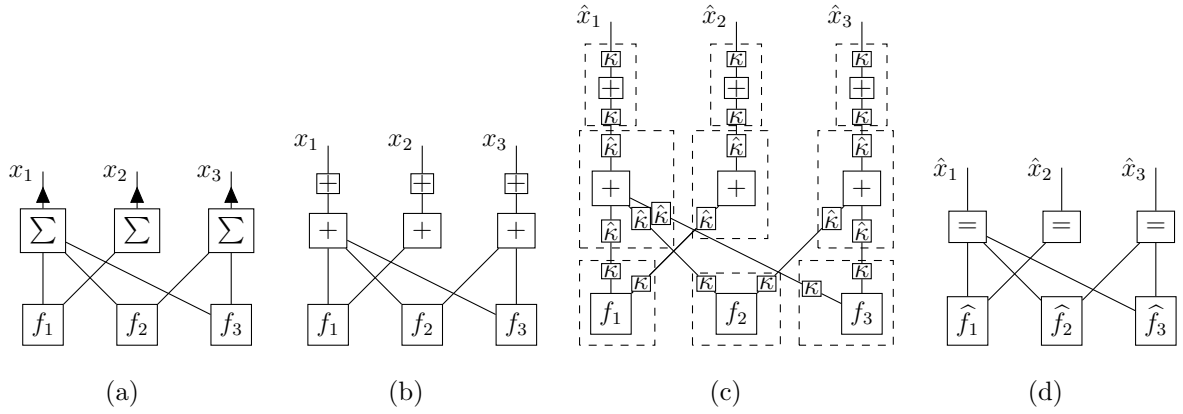


Figure 5.9: Proof of Proposition 4: (a) An example NFG, (b) Step 1, (c) Step 2, and (d) Step 3.

in the following example.

Example 4 *Consider the NFG in Figure 5.8, where the interface functions are taken as sum indicators. Then the exterior function of this NFG is*

$$\sum_{s_1,s_2,s_3} p_{S1S2}(s_1,s_2)p_{S_3}(s_3)\delta_{\sum}(x_1,s_1)\delta_{\sum}(x_2,s_2,s_3) \overset{(a)}{=} \sum_{s_2,s_3} p_{S_1S_2}(x_1,s_2)p_{S3}(s_3)\delta_{\sum}(x_2,s_2,s_3)$$
$$\overset{(b)}{=} p_{S_1S_2}(x_1,x_2) * p_{S3}(x_2),$$

where (a) identifies the bivariate sum indicator with equality indicator and removes it, and (b) is due to Lemma 7. The reader is invited to examine the structure of the original NFG and that of the CFG representing the above convolutional factorization.

Indeed, for any generative NFG model in which interface functions are all sum indicators, the procedure above Proposition 4 applied to an interface function is equivalent to either applying step (a) above (for degree-2 vertices) or applying Lemma 7 (for vertices of degree higher than 2).

It is easy to see that this procedure is reversible, in the sense that one may apply it in reverse direction and convert any CFG to a generative NFG model with all interface functions being sum indicators. Figure 5.10 shows an equivalent pair of CFG and generative NFG model.

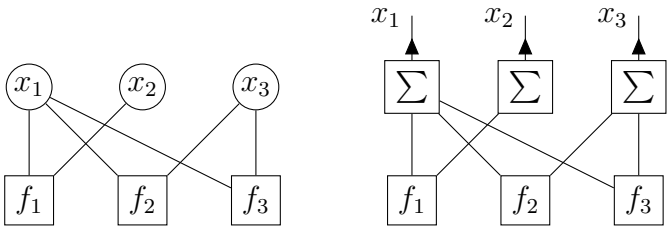


Figure 5.10: An equivalent pair of CFG (left) and generative NFG model (right).

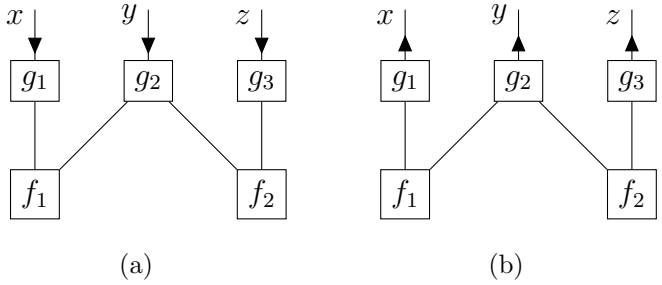


Figure 5.11: (a) $X \perp\!\!\!\perp Z|Y$ and (b) $X \perp\!\!\!\perp Z$.

5.2.5 Independence

We now show that there exists a "duality" between a constrained NFG model and a generative NFG model in their implied independence properties.

Lemma 9 *For the NFG models in Fig. 5.11, we have $X \perp\!\!\!\perp Z|Y$ in the constrained model and $X \perp\!\!\!\perp Z$ in the generative model.*

Proof: For the constrained NFG, it is sufficient to show that $p(x, y, z)$ is a split function via y, see e.g. [39]. To this end, we have g_2 is a split function via y, say it splits into bivariate functions $g_{2,1}$ and $g_{2,2}$. Hence, applying the vertex splitting procedure for g_2 followed by the vertex merging of each hidden function and its adjacent bivariate function, it becomes clear that $p(x, y, z) = f_1'(x, y)f_2'(y, z)$ where $f_1' = \langle f_1, g_1, g_{2,1}\rangle$ and $f_2' = \langle f_2, g_3, g_{2,2}\rangle$.

For the generative model, we prove the claim graphically in Fig. 5.12. Marginalizing y, the probability distribution $p(x, z)$ is realized by the NFG in Fig. 5.12 (a), which by

the definition of a conditional function, is equivalent to the one in (b). Marginalizing again, we obtain the NFGs in (c) and (d) for the marginals $p(x)$ and $p(z)$, respectively. Hence, the multiplication $p(x)p(z)$ is realized by the NFG in (e), which is equivalent (again by the definition of a conditional function) to the one in (f). Noting that the NFG in the left side of (f) realizes the scalar 1, and comparing with (a), we see that $p(x, z)$ and $p(x)p(z)$ are realized by the same NFG, and hence, must be equal. ■

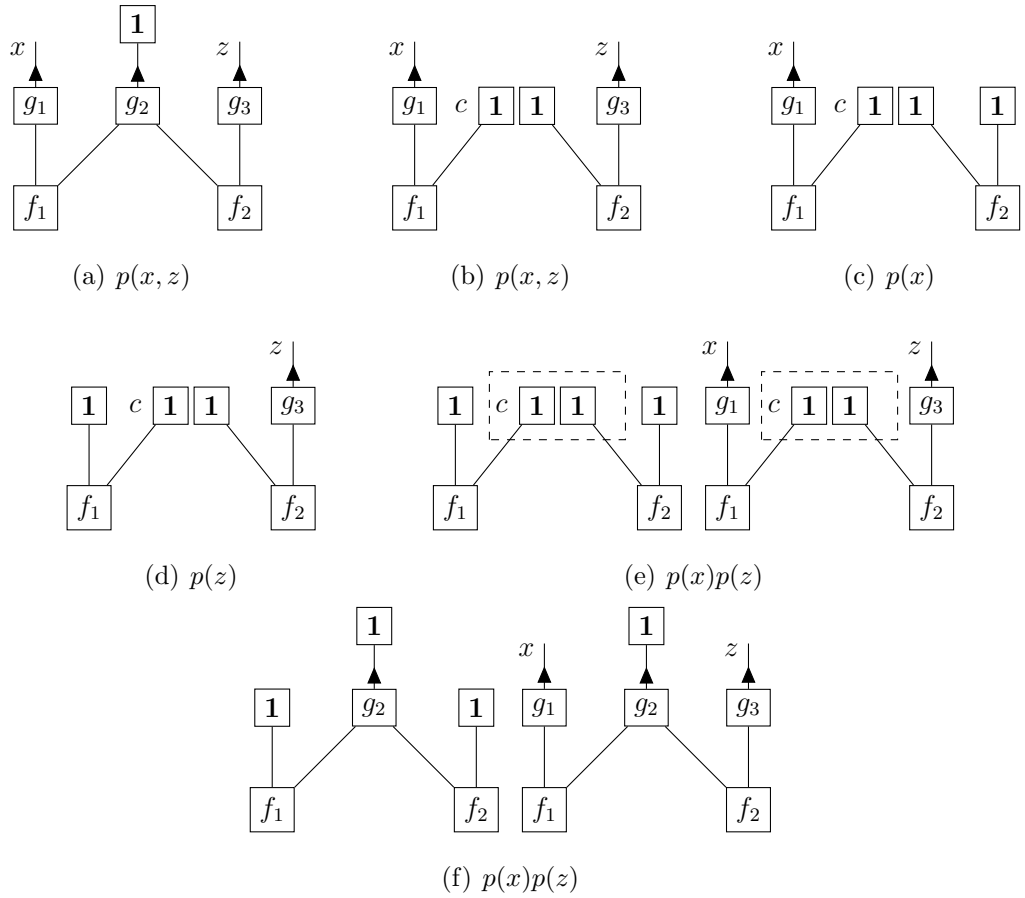


Figure 5.12: Proof of Lemma 9 for the generative model.

We remark that the conditional independence part of the lemma can be proved graphically in a similar manner to the marginal independence part where marginalization is replaced with evaluation. Further, it is interesting to observe that the two NFG models have identical graphs, but the constrained model implies conditional independence property whereas the generative model implies the "dual" marginal independence property.

In an NFG model, suppose that $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{S}$ are three disjoint subsets of vertices. We say $\mathcal{A}$ and $\mathcal{B}$ are *separated* by $\mathcal{S}$ if every path from a vertex in $\mathcal{A}$ to a vertex in $\mathcal{B}$ go through some vertex in $\mathcal{S}$. In this case, if $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{S}$ are all subsets of the interface vertex set I, then, recalling that every external variable is also indexed by the interface vertex it is incident with, we also say that RV sets $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ are separated by the RV set $X_{\mathcal{S}}$. For any subset $I' \subseteq I$, let $\mathrm{ne}(I') := \{\mathrm{ne}(i) : i \in I'\}$. By merging the vertices in $\mathcal{A}$ into one vertex, and similarly for the interface nodes $\mathcal{S}$ and $\mathcal{B}' := I \backslash (\mathcal{A} \cup \mathcal{S})$, and performing the same merging for the hidden nodes $\mathrm{ne}(\mathcal{A})$ and $J \backslash \mathrm{ne}(\mathcal{A})$. Then the resulting NFG has the same graph topology as the ones in Fig. 5.11, as it is clear from the separation property that $\mathrm{ne}(\mathcal{B}') \subseteq J \backslash \mathrm{ne}(\mathcal{A})$. From the fact that the split and conditional properties are preserved under such mergings, the previous lemma extends in a straightforward manner to any NFG model, and we have the following theorem.

Theorem 4 *Let $\mathcal{G}(I \cup J, E, f_{I \cup J})$ be an NFG model and let $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{S}$ be three disjoint interface vertex subsets, i.e., subsets of I. Suppose that $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ are separated by $X_{\mathcal{S}}$. Then*

1. *If the NFG is a constrained NFG model, then $X_{\mathcal{A}} \perp\!\!\!\perp X_{\mathcal{B}} | X_{\mathcal{S}}$.*
2. *If the NFG is a generative NFG model, then $X_{\mathcal{A}} \perp\!\!\!\perp X_{\mathcal{B}}$.*

Part 1 of Theorem 4 is essentially the global Markov property (see, e.g., [39]) on a FG model (noting that constrained NFG models are equivalent to FGs). Part 2 of the theorem, in the special case when all interface functions are sum indicators, was proved in [45] in the context of CFGs (noting that such NFG models reduce to CFGs). That is, Part 2 of Theorem 4 generalizes such a result from CFGs to arbitrary generative NFG models. We now provide some insights for this result.

Consider the NFG in Figure 5.11 (b). The fact $X \perp\!\!\!\perp Z$ can be reasoned by the fact that latent functions f_1 and f_2, giving rise to X and Z respectively, serve as independent sources of randomness. Indeed, it is precisely due to X and Z sharing no common latent functions that when Y is ignored X and Z become independent. The same is true for arbitrary generative NFG models, where if $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ are separated by $X_{\mathcal{S}}$, then we necessarily have $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ share no common latent functions.

We remark that the marginal independence, i.e., Part 2 of Theorem 4 holds for the more general class of NFG models characterized by the property that for each interface function f_i, it holds that

$$\sum_{x_i} f_i(x_i, x_{T(i)}) = \prod_{t \in T(i)} f_t(x_t),$$

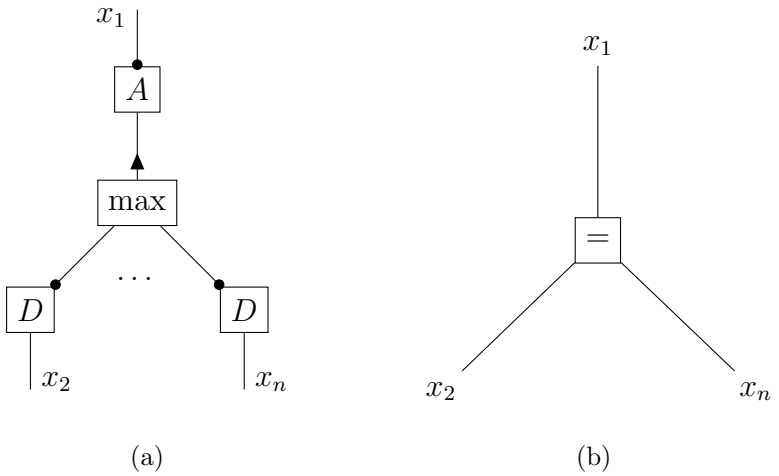
for some univariate functions f_t. We may refer to an NFG model whose interface functions satisfy this property as an *extended generative model*, and it is clear that a generative model is an extended generative model. (A conditional function trivially satisfies the property above.) It is not hard to show that the class of constrained models and the class of extended generative models are closed under internal holographic transformations, from which, it follows that the independence properties in Theorem 4 are invariant under internal holographic transformations.

5.3 Transformed NFG models

In some applications, instead of modelling the joint probability distribution of the RVs, we may wish our model to represent a certain transformation of the joint distribution, and it becomes clear that the NFG modelling framework introduced in this work is particularly convenient for this purpose. In subsequent discussions, a *transformed NFG model*, or simply a transformed model, refers to any NFG obtained from an NFG model (generative or constrained) by a holographic transformation, where the external transformers of the holographic transformation are not necessarily trivial. At some places we may refer to the original NFG as the *base model*.

Next we show that a particular class of NFG models, upon an appropriate choice of holographic transformation, results in CDNs. Let $\mathcal{X} := \{1, \ldots, |\mathcal{X}|\}$ and let bivariate function $A_{\mathcal{X}}$ on $\mathcal{X} \times \mathcal{X}$ be such that $A_{\mathcal{X}}(x, x') = 1$ if $x' \leq x$ and $A_{\mathcal{X}}(x, x') = 0$, otherwise. We call $A_{\mathcal{X}}$ a *cumulus* function. Let bivariate function $D_{\mathcal{X}}$ on $\mathcal{X} \times \mathcal{X}$ be such that $D_{\mathcal{X}}(x, x') = 1$ if $x = x'$, $D_{\mathcal{X}}(x, x') = -1$ if $x = x' + 1$, and $D_{\mathcal{X}}(x, x') = 0$ otherwise. We call $D_{\mathcal{X}}$ a *difference* function.

In the case where $\mathcal{X}$ is the Cartesian product $\prod_{i \in I} \mathcal{X}_i$ where $\mathcal{X}_i := \{1, \ldots, |\mathcal{X}_i|\}$ and I is a finite indexing set, then the previous definitions are extended to the partially-



(a) (b)

Figure 5.13: Two equivalent NFGs

ordered set $\mathcal{X}$ in a component-wise manner by setting $A_{\mathcal{X}}(x_I, x'_I) := \prod_{i\in I} A_{\mathcal{X}_i}(x_i, x'_i)$ and $D_{\mathcal{X}}(x_I, x'_I) := \prod_{i\in I} D_{\mathcal{X}_i}(x_i, x'_i)$ for all $x_I, x'_I \in \mathcal{X}$. As in Chapter 4, in our notations for cumulus and difference function vertices, we distinguish the first argument using a dot to mark the corresponding edge, cf. Fig. 5.13 (a).

The following lemma will be useful in characterizing CDNs as a subclass of transformed NFG models.

Lemma 10 *Let* $\mathcal{X} = \prod_{i\in I} \mathcal{X}_i$ *where* $\mathcal{X}_i := \{1, \ldots, |\mathcal{X}_i|\}$ *for all* $i \in I$. *Then,*

1. $\left\langle A_{\mathcal{X}}(x, y), D_{\mathcal{X}}(y, x')\right\rangle = \delta_{=}(x, x')$.

2. For any set of RVs X_I, $\left\langle p_{X_I}(x'), A_{\mathcal{X}}(x, x')\right\rangle$ *is* $F_{X_I}(x)$ *for all* $x \in \mathcal{X}$.

3. The two NFGs in Figure 5.13 are equivalent where each edge variable assumes its values from $\mathcal{X}$.

Proof: Parts 1 and 2 are immediate from the definitions of $A_{\mathcal{X}}$ and $D_{\mathcal{X}}$. For part 3, let $\mathcal{G}$ be as in Fig. 5.13 (a). First we prove the result for $n = 3$ and $|I| = 1$, i.e., $\mathcal{X} = \{1, \ldots, |\mathcal{X}|\}$, where by the definitions of the difference transform, the exterior

function, and the max indicator function, we have $Z_{\mathcal{G}}(x_1, x_2, x_3)$ is given by

$$Z_{\mathcal{G}} = \begin{cases} A_{\mathcal{X}}(x_1, \max(x_2, x_3)) - A_{\mathcal{X}}(x_1, \max(x_2 + 1, x_3)) - \\ A_{\mathcal{X}}(x_1, \max(x_2, x_3 + 1)) + A_{\mathcal{X}}(x_1, \max(x_2 + 1, x_3 + 1)), & x_2 < |\mathcal{X}|, x_3 < |\mathcal{X}|, \\ A_{\mathcal{X}}(x_1, \max(x_2, x_3)) - A_{\mathcal{X}}(x_1, \max(x_2 + 1, x_3)), & x_2 < |\mathcal{X}|, x_3 = |\mathcal{X}|, \\ A_{\mathcal{X}}(x_1, \max(x_2, x_3)) - A_{\mathcal{X}}(x_1, \max(x_2, x_3 + 1)), & x_2 = |\mathcal{X}|, x_3 < |\mathcal{X}|, \\ A_{\mathcal{X}}(x_1, \max(x_2, x_3)), & x_2 = x_3 = |\mathcal{X}|. \end{cases}$$

From the definition of the cumulus, it is clear that any possible non-zero values of $Z_{\mathcal{G}}$ may occur only if $x_1 \geq \max(x_2, x_3)$. Assume $x_1 > \max(x_2, x_3)$ and note that in this case it is impossible to simultaneously have $x_2 = |\mathcal{X}|$ and $x_3 = |\mathcal{X}|$, then

$$\begin{aligned} Z_{\mathcal{G}}(x_1, x_2, x_3) &= \begin{cases} 1 - 1 - 1 + 1, & x_2 < |\mathcal{X}|, x_3 < |\mathcal{X}|, \\ 1 - 1, & x_2 < |\mathcal{X}|, x_3 = |\mathcal{X}|, \\ 1 - 1, & x_2 = |\mathcal{X}|, x_3 < |\mathcal{X}|, \end{cases} \\ &= 0. \end{aligned}$$

Hence, assume $x_1 = \max(x_2, x_3)$ and further suppose $x_2 < x_3$, then

$$\begin{aligned} Z_{\mathcal{G}}(x_1, x_2, x_3) &= \begin{cases} 1 - 1, & x_2 < |\mathcal{X}|, x_3 < |\mathcal{X}|, \\ 1 - 1, & x_2 < |\mathcal{X}|, x_3 = |\mathcal{X}|, \end{cases} \\ &= 0. \end{aligned}$$

By symmetry, we also have $Z_{\mathcal{G}}(x_1, x_2, x_3) = 0$ for $x_2 > x_3$. The only possibility left is $x_2 = x_3$, in which case, it is clear that $Z_{\mathcal{G}}(x_1, x_2, x_3) = 1$. For $|I| > 1$, the claim follows from the fact that the cumulus, difference, max and equality functions are defined in a component-wise manner.

For general n, using the fact that $\max(x_2, x_3, \ldots, x_n) = \max(x_2, \max(x_3, \ldots, \max(x_n - 1, x_n) \ldots))$, we can express the NFG in Fig. 5.13 (a) using the NFG in Fig. 5.14 (a). Inserting the inverse-pair $A_{\mathcal{X}}$ and $D_{\mathcal{X}}$ on each edge inside the dashed box in Fig. 5.14 (a), we obtain the equivalent NFG in (b). Invoking the established part of the lemma for $n = 3$ under the vertex merging shown in (b), and the claim follows. ■

In this lemma, Part 1 suggests that $A_{\mathcal{X}}$ and $D_{\mathcal{X}}$ are an inverse-pair transformers, and Part 2 suggests that cumulus functions may serve to transform a probability distribution to a CDF.

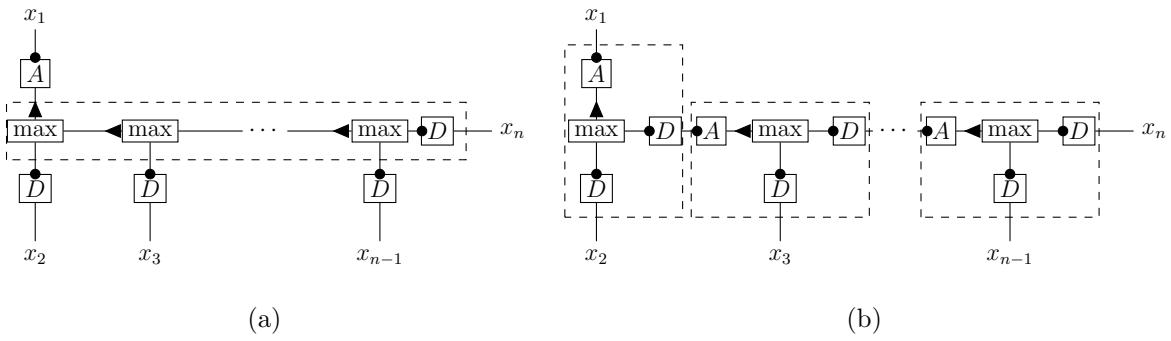


Figure 5.14: Proof of Part 3 of Lemma 10 for arbitrary n.

Given a generative NFG in which each interface function is a max indicator and each hidden function is a probability distribution of the variables incident on it. Let $\mathcal{G}$ be the transformed model obtained from such generative NFG by inserting the cumulus transformer $A_{\mathcal{X}_i}$ on each half-edge. The following procedure converts $\mathcal{G}$ into a CDN:

1) Replace each max indicator and its adjacent transformer with a variable node representing the transformer's half-edge variable, and delete the half edge.

2) Replace each hidden function, which is a probability distribution, with the corresponding cumulative distribution.

Theorem 5 *If in a transformed model all external transformers are cumulus transformers, all interface functions are max indicators, and all hidden functions are probability distributions, then the above procedure gives rise to a CDN equivalent to the transformed model.*

Proof: Perform a holographic transformation on the transformed model by inserting into each internal edge e the inverse-pair transformers $A_{\mathcal{X}_e}$ and $D_{\mathcal{X}_e}$, with the cumulus facing the hidden function and the difference transformer facing the max indicator. Merging each hidden node with its neighboring cumulus transformers, by Part (2) of Lemma 10, the resulting node represents the desired CDF. Merging each max indicator with its neighboring difference transformers and the already existing cumulus, by Part (3) of Lemma 10, we arrive to a constrained NFG in which each interface function is an equality indicator, and the claim follows by Proposition 2. ■

Figure 5.15 demonstrates the proof on an example NFG.

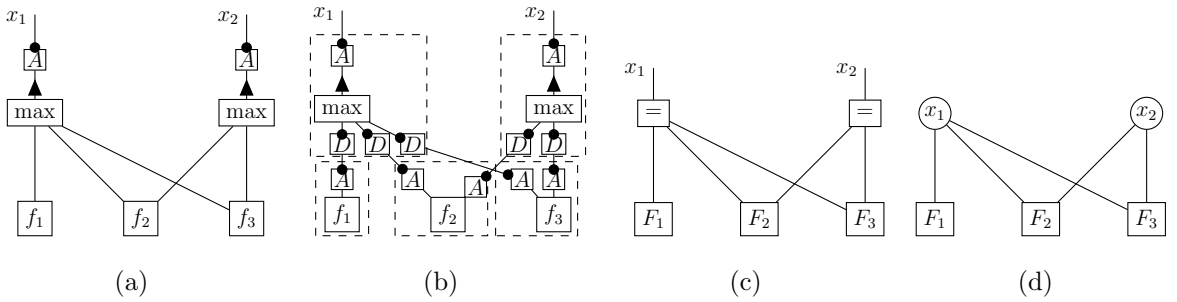


Figure 5.15: (a) An example of a transformed model in accordance to Theorem 5. (b) Inserting the inverse-pair cumulus-difference transformers on each internal edge, and vertex merging each node with its neighboring transformers, by the GHT, the resulting NFG is equivalent to the one in (a). (c) By Lemma 10 an equivalent NFG to the one in (b), where F_i is the cumulus transform of f_i, which is a CDF if f_i is a probability distribution. (d) By Proposition 2, an equivalent CDN to the NFG in (c).

Before we proceed, we remark that the independence properties implied by a transformed model are precisely the ones implied by its base NFG, i.e., by ignoring all the external transformers. By the remark succeeding Theorem 4, such independence properties are further invariant under internal holographic transformations. Since the base NFG of the transformed model in Theorem 5 is a generative model, it becomes clear that the independence properties implied by CDNs are the marginal independence ones, i.e., Part 2 of Theorem 4.

Besides CDNs, we also remark that it is possible to regard linear characteristic models (LCMs) [6] as a special case of transformed NFG models. In fact, the NFG in Fig. 5.9 (d) is a transformed model that is equivalent to an LCM, and the holographic transformation presented in the proof of Proposition 4, Fig. 5.9 (c), provides the basis for understanding an LCM as a transformed model whose base NFG is a generative NFG equivalent to a CFG, Fig. 5.9 (a). We skip details, and hope the framework of transformed NFG models is clear enough to see such equivalence.

5.4 Linear codes

In this section we take a pause and discuss the connection between NFG models and linear codes. The material in this section is well-known [24], yet, we choose to address it in the light of the constrained and generative semantics.

Any code $\mathcal{C}$, linear or not linear, can be described in terms of its *membership* function, namely, the indicator function $\delta_{\mathcal{C}}(y) := [y \in \mathcal{C}]$ for all $y \in \mathcal{X}$ and for some finite set $\mathcal{X}$. Clearly $\delta_{\mathcal{C}}$ may be viewed, up to scaling factor, as a distribution function over $\mathcal{X}$ and in the case of linear codes, as we will see, may be described in terms of a constrained or a generative NFG model.

A *linear code* of length n and dimension k over a finite field $\mathbb{F}$ is a k dimensional subvector space of $\mathbb{F}^n$. Classically, a linear code $\mathcal{C}$ can be expressed in two dual ways. At one hand, $\mathcal{C} = \{f(x) : x \in \mathbb{F}^k\}$ for some linear function $f : \mathbb{F}^k \to \mathbb{F}^n$. This can also be written as $\mathcal{C} = \{\big(f_1(x), \ldots, f_n(x)\big) : x \in \mathbb{F}^k\}$ for some linear maps $f_i : \mathbb{F}^k \to \mathbb{F}$ for all i. That is, the code indicator function $\delta_{\mathcal{C}}(y)$ for all $y \in \mathbb{F}^n$, can be expressed as the sum-of-products form

$$\delta_{\mathcal{C}}(y_1, \ldots, y_n) = \sum_{x \in \mathbb{F}^k} \prod_{i=1}^{n} [y_i = f_i(x)],$$

for all $y_i \in \mathbb{F}$. Clearly, each local function $[y_i = f_i(x)]$ is a conditional function of y_i given x, and so, the indicator function of $\mathcal{C}$ can be realized using a generative NFG model. In fact, since each linear function $f_i : \mathbb{F}^k \to \mathbb{F}$ can be written as $f_i(x_1, \ldots, x_k) = a_{i1}x_1 + \cdots + a_{ik}x_k$ for some $a_{ij} \in \mathbb{F}$, it follows that each interface function is a sum indicator function involving y_i and x_j for all j such that $a_{ij} \neq 0$. The role of the hidden functions is to provide replicas of each variable x_j for all $j \in \{1, \ldots, k\}$, and hence are taken as equality indicators. More explicitly, for each x_j, we have a hidden equality indicator of degree equal to the number of interface functions involving x_j. This guarantees that each variable appears in the desired number of interface functions while respecting the degree restrictions [23]. From this we arrive to a generative NFG with n interface nodes (each is a sum indicator), k hidden nodes (each is an equality indicator), and there is an edge connecting nodes i and j if and only if a_{ij} is nonzero. Fig. 5.16 (a) illustrates the generative NFG model for the Hamming code (with $n = 7$ and $k = 4$).

On the other hand, the parity check interpretation of a linear code dictates that the elements of a linear code $\mathcal{C}$ must satisfy a collection of homogeneous linear equations. That is, $\mathcal{C} = \{y \in \mathbb{F}^n : f(y) = 0\}$ for some linear map $f : \mathbb{F}^n \to \mathbb{F}^{n-k}$. This can also be written as $\mathcal{C} = \{y \in \mathbb{F}^n : (f_1(y), \ldots, f_{n-k}(y)) = 0\}$ for some linear maps $f_j : \mathbb{F}^n \to \mathbb{F}$. Hence, the code indicator function can be expressed as the product

$$\delta_{\mathcal{C}}(y) = \prod_j [f_j(y) = 0],$$

which (since f_j is linear) can further be simplified as $\delta_{\mathcal{C}}(y_1, \ldots, y_n) = \prod_j [\sum_i a_{ij} y_i = 0]$ for some $a_{ij} \in \mathbb{F}$. From Proposition 2, we can see that the code indicator function is realized by a constrained NFG model in which each interface node i is an equality indicator, each hidden node j is a parity indicator, and there is an edge connecting nodes i and j if and only if a_{ij} is nonzero, Fig. 5.17 (a).

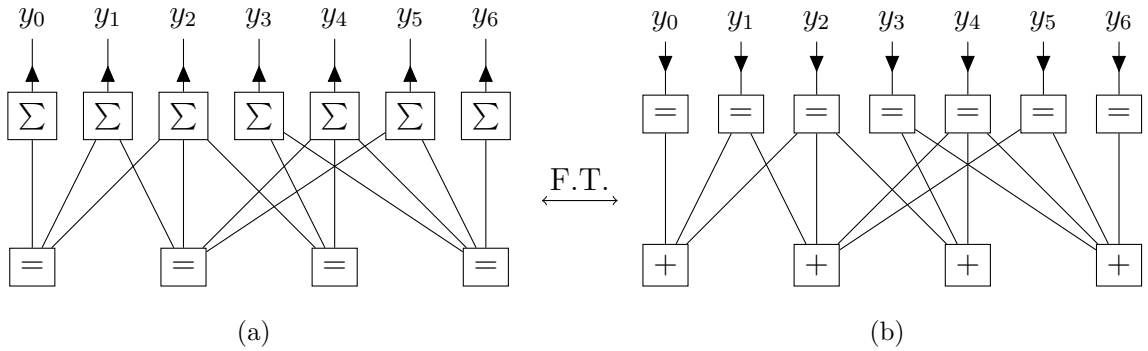


Figure 5.16: (a) Generator realization of Hamming code, and (b) parity realization of dual Hamming code.

In summary, a constrained NFG model of a linear code represents a parity realization, and a generative NFG model represents a generator realization. From the duality between the sum indicator and the equality indicator under the Fourier transform, the duality between a generator realization of a code and a parity realization of the dual code may be explained as follows: Starting with a generative NFG model (generator realization) of a linear code, Fig. 5.16 (a), and performing a holographic transformation with Fourier transformers, one obtains a constrained NFG model (a parity realization) of the dual code, Fig. 5.16 (b). Conversely, starting with a constrained NFG model (a

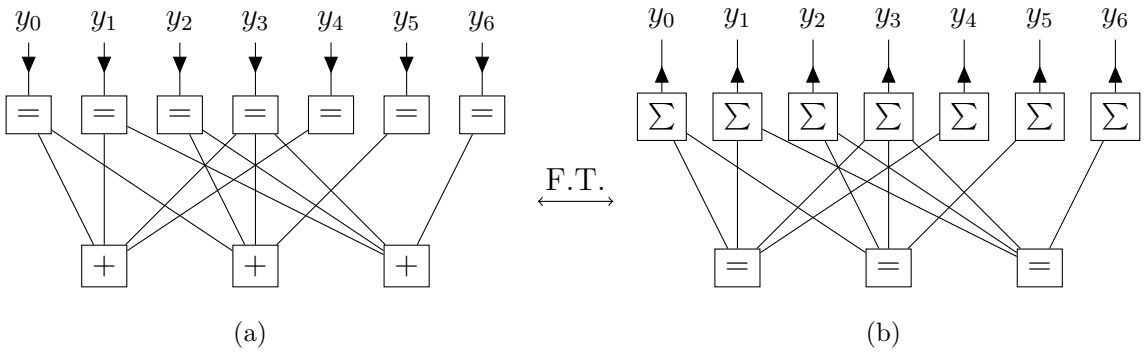


Figure 5.17: (a) Parity realization of Hamming code, and (b) generator realization of dual Hamming code.

parity realization) of the code, Fig. 5.17 (a), one ends with a generative NFG model (a generator realization) of the dual code, Fig. 5.17 (b).

5.5 Evaluation of the exterior function

In this section we discuss the algorithmic aspect of evaluating the exterior function of NFGs. We start with the "elimination algorithm" for NFGs, which is essentially the well-known elimination algorithm of inference on undirected graphical models [32]. The elimination algorithm is exact but its complexity depends on the ordering at which the elimination is performed.[3] A more efficient algorithm, but exact only on NFGs with no cycles, is the sum-product algorithm [38], which we discuss in the language of NFGs [26] in the second part of this section. Finally, we discuss an "indirect approach" for evaluating the exterior function, where a holographic transformation, if it exists, is used to convert the NFG into one that is more appropriate for the such computation.

[3]The problem of finding the "best" elimination ordering is known to be NP-hard, where the term "best" is in the sense of minimizing the largest node-degree (of nodes that do not factor multiplicatively) arising while performing the elimination algorithm, and the minimization is over all possible orderings.

5.5.1 Elimination algorithm

The following algorithm computes the exterior function of an arbitrary NFG (i.e. not necessarily a bipartite).

Algorithm 1 (Elimination) *Given an NFG $\mathcal{G}$.*

`While` $\mathcal{G}$ `is not a single node. Do`

{

`Pick an adjacent pair of vertices` v_1 `and` v_2 `in` $\mathcal{G}$`;`

`Compute` $f_{v_1v_2}(x_{E(v_1)\cup E(v_2)\setminus E(v_1)\cap E(v_2)}) := \langle f_{v_1}, f_{v_2}\rangle$;

`Update` $\mathcal{G}$ `by removing` f_{v_1} `and` f_{v_2}`, and adding` $f_{v_1v_2}$`;`

}

Evidently, this algorithm runs in a finite time and terminates with a single node whose function is the desired exterior function. This is essentially the vertex merging procedure applied recursively on pairs of adjacent vertices. Clearly, the elimination algorithm may equivalently be viewed as an elimination algorithm on the edges of the NFG, where in each step it eliminates all the edges between a pair of adjacent vertices. More precisely, one may say, the elimination algorithm is a merging algorithm on the nodes, and is an elimination algorithm on the edges of the NFG. In subsequent discussions, we will freely alternate between such two views. Note that even if we start with a simple NFG, parallel edges may still arise[4] while applying the elimination algorithm. However, loops may never arise since in each step we eliminate all parallel edges at once.

Example 5 *Let $\mathcal{G}$ be as in Fig 5.18 (a). Applying the elimination algorithm, we obtain: Eliminating y_1 gives the NFG in Fig. (b) with*

$$f_{12}(x_1, x_2, y_2, y_3) = \big\langle f_1(x_1, y_1, y_3), f_2(x_2, y_1, y_2)\big\rangle,$$

eliminating y_2, y_3 gives the NFG in Fig. (c) with

$$f_{123}(x_1, x_2, x_3) = \big\langle f_{12}(x_1, x_2, y_2, y_3), f_3(x_3, y_2, y_3)\big\rangle,$$

and it is easy to verify that $Z_{\mathcal{G}} = f_{123}$.

[4]It is not hard to see that parallel edges do not appear at any step of the elimination algorithm if and only if the NFG is cycle-free, i.e., is a tree.

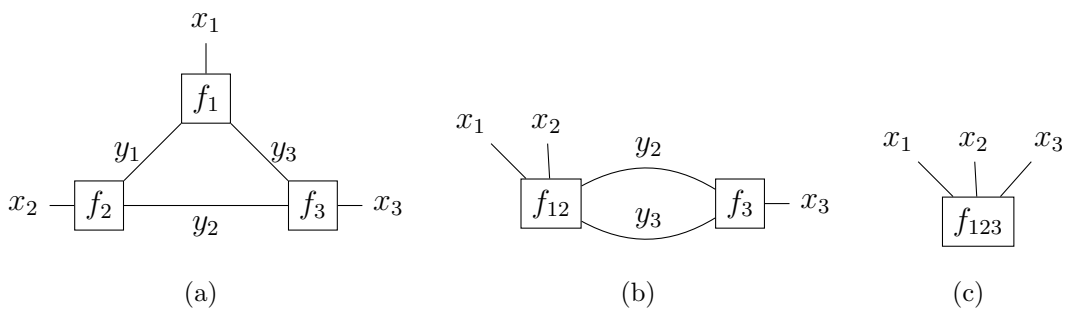


Figure 5.18: Elimination algorithm example.

For any node v in an NFG, let $\deg(v) := |E(v)|$ be the number of external and internal edges incident on v. The following lemma determines the complexity of eliminating a pair of adjacent vertices in an NFG.

Lemma 11 *The complexity of eliminating a pair of adjacent vertices u, v in the elimination algorithm is of order $|\mathcal{X}|^{\deg(u)+\deg(v)-|E(u)\cap E(v)|}$.*

Proof: For each $x \in \mathcal{X}_{E(u)\cup E(v)\backslash E(u)\cap E(v)}$, we need $|\mathcal{X}_{E(u)\cap E(v)}|$ computations, and the claim follows by noting that $|E(u)\cup E(v)\backslash E(u)\cap E(v)| = |E(u)|+|E(v)|-2|E(u)\cap E(v)|$. ■

The following example shows that for some indicator functions of interest, the elimination complexity can significantly be reduced.

Example 6 *Consider the NFG in Fig. 5.19 (a), where each edge is associated the same alphabet $\mathcal{X}$. In general, the complexity of computing the exterior function is of order $|\mathcal{X}|^{n+1}$. In the special case where f is the indicator function $\delta_{\sum}$ or $\delta_{\max}$, then the complexity is $(n-1)|\mathcal{X}|^2$. This is because such indicator functions may further be factorized as shown in Figs. 5.19 (b) and (c). To see this, the elimination algorithm may start by eliminating $t_1, \ldots, t_n$, which induces no complexity as each elimination simply accounts to pointing to the proper entry of each function f_i, resulting in Fig. (d), where f_i' is properly defined according to $\delta_{\sum}$ or $\delta_{\max}$. Now eliminating each e_i (starting with e_{n-1}) costs $|\mathcal{X}|^2$ computations, giving rise to $(n-1)|\mathcal{X}|^2$ computations in total. Note that if $f = \delta_{=}$, then the complexity of computing the exterior function is $(n-1)|\mathcal{X}|$. (For each $x \in \mathcal{X}$, we need $(n-1)$ multiplications.)*

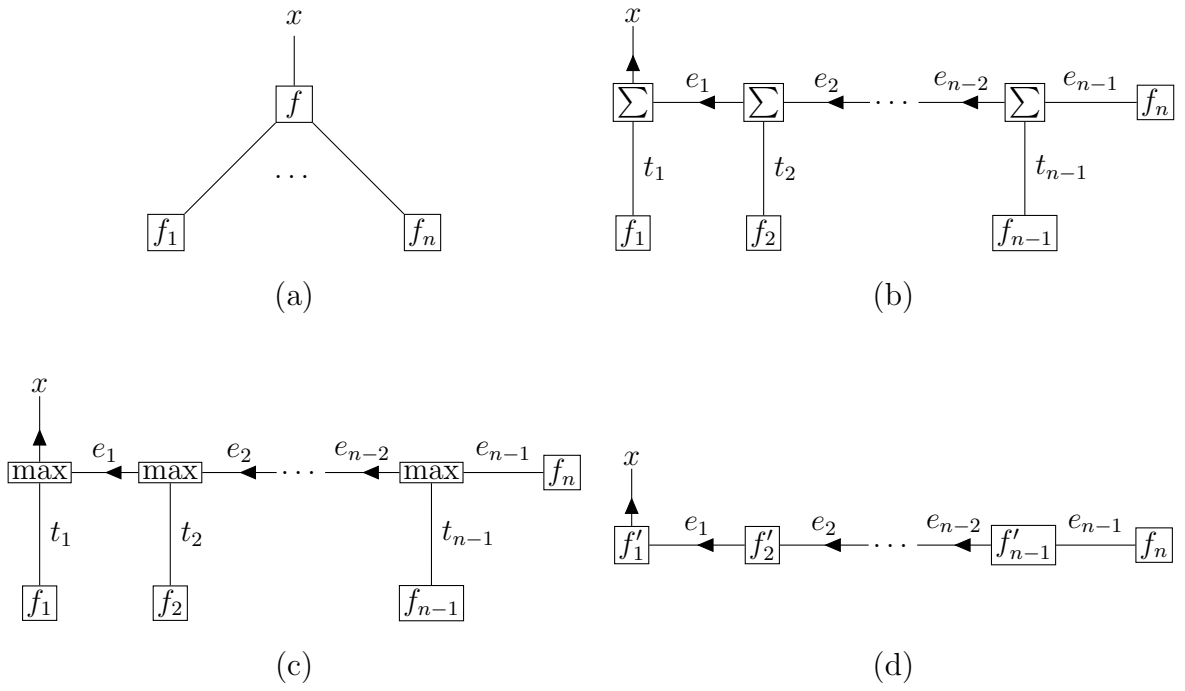


Figure 5.19: Example 6: The complexity of eliminating a sum or max indicator function.

It is not hard to see that one may impose an elimination ordering such that the elimination algorithm may be viewed as one that merges a node with its neighbors, picks another node and merges it with its neighbors, and so on, until there is only one node left. We refer to such elimination as "block elimination," and to the elimination algorithm at the block level as the "block elimination algorithm," i.e., in each step, the block elimination algorithm replaces a function f_i and its neighbors with the function $\big\langle f_i, f_j : j \in \mathrm{ne}(i)\big\rangle$. In the special case where each block eliminated function f_i is the equality indicator and none of the neighbors of f_i share any edges, cf. Fig. 5.20 and Example 7, then the block elimination becomes the classical elimination algorithm of undirected graphical models. (An undirected graphical model can be converted to a FG which by Proposition 2 is equivalent to a constrained NFG whose interface functions are equality indicators. Note that none of the neighbors of such equality nodes shares any edges due to the bipartite nature of an NFG model.) We remark that whereas the elimination algorithm does not preserve the bipartite structure of an NFG in each edge elimination step, on the block elimination level, the algorithm respects such bipartite

structure. (Let I and J be the two independent vertex sets, and let $f_{\mathrm{ne}(i)}$ be the node resulting from merging node $i \in I$ and its neighbors. Let node $j \in J$ be connected to $f_{\mathrm{ne}(i)}$ by some edge e, then $e \in E(i)$ or $e \in E(j')$ for some $j' \in \mathrm{ne}(i)$. Both such cases are impossible, since $e \in E(i)$ implies $j \in \mathrm{ne}(i)$ and so j would have been merged with i in the block elimination, and $e \in E(j')$ violates the bipartite assumption.)

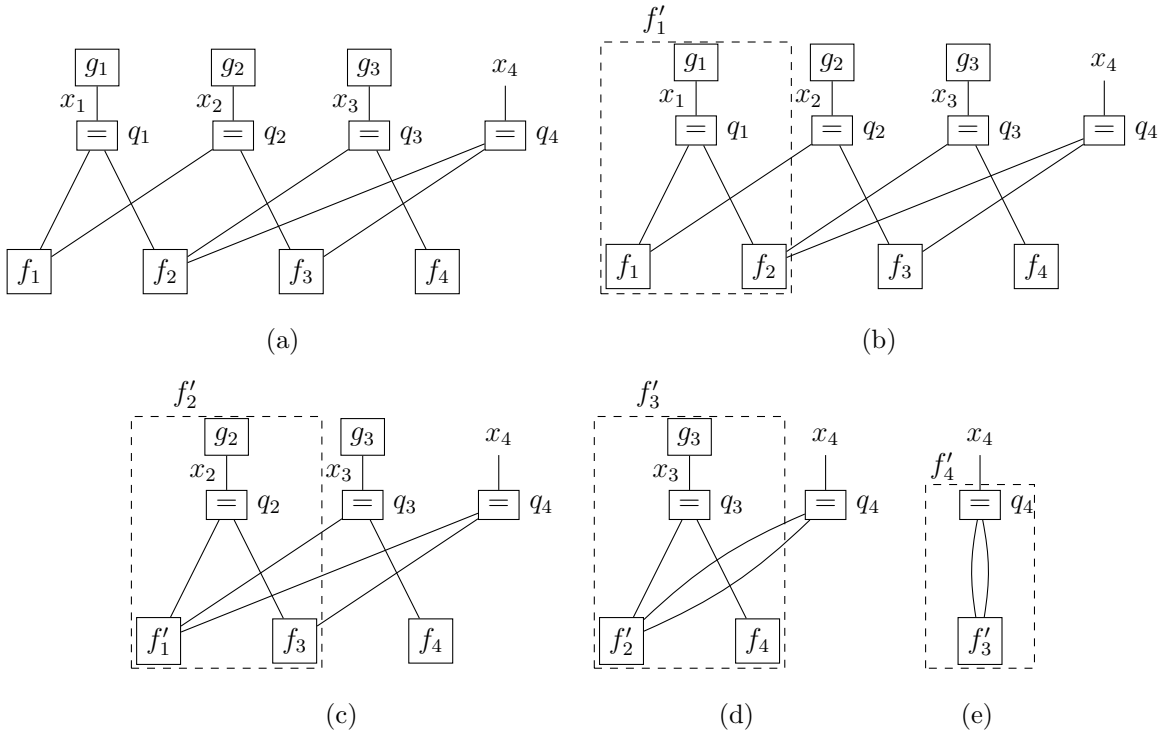


Figure 5.20: Block elimination algorithm: (a) an example NFG $\mathcal{G}$, (b), (c), (d), and (e) illustrate the block elimination of nodes q_1, q_2, q_3, and q_4, respectively, and it is not hard to see that $Z_{\mathcal{G}}(x_4) = f_4'(x_4) = f_3'(x_4, x_4)$. Note that if $\mathcal{G}$ is obtained from a constrained NFG by gluing functions g_1, g_2, and g_3 into the external edges x_1, x_2, and x_3 of the constrained NFG, then in the special case where every g_i is the constant-one indicator, it follows that $Z_{\mathcal{G}}$ is the marginal probability distribution p_{X_4}, cf. Section 5.6.

The following example addresses the complexity of a block elimination.

Example 7 *Let $\mathcal{G}$ be as in Fig. 5.21, where $E_1, \ldots, E_n$ are disjoint. This NFG may be understood as a sub-NFG, the computation of its exterior function corresponds to a block elimination of a node and its neighbors in a bigger NFG. Note that if the bigger*

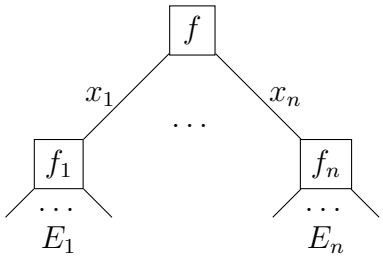


Figure 5.21: Example 7.

NFG is a bipartite, then the requirement that $E_1, \ldots, E_n$ be disjoint is automatically satisfied. Assuming the variable of each edge incident on f takes its values from $\mathcal{X}$, then by recursive application of Lemma 11, the complexity of computing the exterior function $Z_{\mathcal{G}}(x_{E_1}, \ldots, x_{E_n})$ is given by (assuming the elimination order $x_1, x_2, \ldots, x_n$)

$$\sum_{i=1}^{n} |\mathcal{X}|^{1+n-i+\sum_{j=1}^{i}|E_j|}.$$

That is, if E_i is non-empty for all i, then the complexity is of order $|\mathcal{X}|^{1+|E_1|+\cdots+|E_n|}$. As an example, consider for instance $n = 3$, then the exterior function of $\mathcal{G}$ is given by

$$Z_{\mathcal{G}}(x_{E_1}, x_{E_2}, x_{E_3}) = \overbrace{\Big\langle f_3, \big\langle f_2, \underbrace{\underbrace{\langle f_1, f\rangle}_{\text{SP1}}}_{\text{SP2}} \big\rangle \Big\rangle}^{\text{SP3}}.$$

Computing the first sum of products (SP1), i.e., eliminating x_1, involves $|\mathcal{X}|^{|E_1|+3}$ computations by Lemma 11, and similarly, one needs $|\mathcal{X}|^{|E_1|+|E_2|+2}$ operations to compute SP2. Finally, SP3 requires $|\mathcal{X}|^{|E_1|+|E_2|+|E_3|+1}$ operations.

Next we consider the transformation of a function. Suppose we are interested in the exterior function of the NFG in Fig. 5.22 (a), where x and x' take their values from a finite alphabet $\mathcal{X}^n$ for some positive integer n. Clearly the exterior function represents a matrix-vector multiplication, and hence may be computed in $|\mathcal{X}|^{2n}$ operations. In the case where the bivariate function (on $\mathcal{X}^n \times \mathcal{X}^n$) f' factors as the product of bivariate functions (on $\mathcal{X} \times \mathcal{X}$) $f'_1, \ldots, f'_n$, then from the previous example, it follows that the complexity of transforming the function f via transformers $f'_1, \ldots, f'_n$ is $n|\mathcal{X}|^{n+1}$. (This

is the special case with $|E_1| = \cdots = |E_n| = 1$, compare Figs 5.21 and 5.22 (b).) Another way of viewing this is that the elimination of each x_i accounts to a matrix-vector multiplication for a given configuration $x_{N\setminus\{i\}}$, where $N := \{1, \ldots, n\}$, and hence requires $|\mathcal{X}|^2$ computations. That is, eliminating x_i requires $|\mathcal{X}|^{(n-1)+2}$ operations, and eliminating $x_1, \ldots, x_n$ requires in total $n|\mathcal{X}|^{n+1}$ operations, as observed.

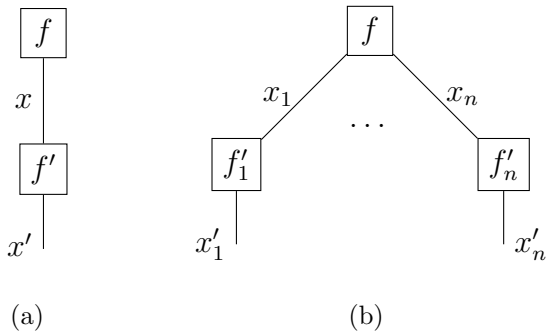


Figure 5.22: Transformation of a function.

We emphasize that if the transformers exhibit a special form, then more efficient computations may be achieved. For instance, if each f'_i is a Fourier kernel, then the fast Fourier transform may be used in the elimination of each edge using $\log(|\mathcal{X}|)|\mathcal{X}|^n$ computations, giving rise to a total $n\log(|\mathcal{X}|)|\mathcal{X}|^n$ operations for computing the transformation of f. Another important case where such savings are possible is when each f'_i is the cumulus or the difference transform. In such case, the matrix-vector multiplication associated with the elimination of each edge may be performed using $|\mathcal{X}|$ operations, giving rise to a total complexity of $n|\mathcal{X}|^n$. To see this, consider our example with $n = 3$ and assume f'_1, f'_2, f'_3 are cumulus transforms. To eliminate x_1, we start with the initiation $s := f$, and for $x'_1 = 2, \ldots, |\mathcal{X}|$, we update s as,

$$s(x'_1, x_2, x_3) = f(x'_1, x_2, x_3) + s(x'_1 - 1, x_2, x_3).$$

Hence, computing SP1 in $|\mathcal{X}|^3$ operations instead of $|\mathcal{X}|^4$. Similarly, we compute SP2 and SP3, giving rise to a total number of operations $3|\mathcal{X}|^3$. This clearly extends to any n, and below we provide two algorithms for fast computation of the cumulus and difference transformations, where to facilitate notation, we use J^- to denote the set $\{1, \ldots, j-1\}$ and J^+ to denote $\{j+1, \ldots, n\}$ for any $j \in \{1, \ldots, n\}$.

Algorithm 2 (Fast cumulus transform) `Initialize:` $s_0 = f$;
`For` $j = 1, \ldots, n\{$
  `For each` $(x'_{J^-}, x_{J^+}) \in \mathcal{X}^{n-1}\{$
    `Initialize:` $s_j(x'_{J^-}, x'_j = 1, x_{J^+}) = s_{j-1}(x'_{J^-}, x_j = 1, x_{J^+})$;
    `For` $x'_j = 2, \ldots, |\mathcal{X}|\{$
    $s_j(x'_{J^-}, x'_j, x_{J^+}) = s_{j-1}(x'_{J^-}, x'_j, x_{J^+}) + s_j(x'_{J^-}, x'_j - 1, x_{J^+})$;
    $\}$
  $\}$
$\}$
`Return` s_n;

The difference transformation is performed in exactly the same manner, except for the updating rule:

Algorithm 3 (Fast difference transform) `Initialize:` $s_0 = f$;
`For` $j = 1, \ldots, n\{$
  `For each` $(x'_{J^-}, x_{J^+}) \in \mathcal{X}^{n-1}\{$
    `Initialize:` $s_j(x'_{J^-}, x'_j = 1, x_{J^+}) = s_{j-1}(x'_{J^-}, x_j = 1, x_{J^+})$;
    `For` $x'_j = 2, \ldots, |\mathcal{X}|\{$
    $s_j(x'_{J^-}, x'_j, x_{J^+}) = s_{j-1}(x'_{J^-}, x'_j, x_{J^+}) - s_{j-1}(x'_{J^-}, x'_j - 1, x_{J^+})$;
    $\}$
  $\}$
$\}$
`Return` s_n;

5.5.2 Sum-product algorithm

Given an NFG $\mathcal{G}$ with no external edges, in many circumstances, for each edge e in the NFG, one may be interested in computing the *marginal exterior function* [26] defined as

$$Z_{\mathcal{G}}(x_e) := \sum_{x_{E \setminus \{e\}}} \prod_{v \in V} f_v(x_{E(v)}).$$

It is possible to compute such marginals using the elimination algorithm by imposing an elimination ordering such that x_e appears last. (More precisely, the elimination algorithm is slightly modified here such that when it reaches x_e it multiplies the functions of the two nodes incident with e without summing out x_e.) Although the elimination algorithm is exact for any NFG, it may be expensive to perform, as it is likely to produce nodes with large degrees. Further, in addressing the marginals problem above, the elimination algorithm is repeated for each marginal, giving rise to some redundant computations since most of the computations used for evaluating one of the marginals can be used in determining some other marginals. The *sum-product algorithm* (SPA) is an efficient alternative in the case of NFGs with no cycles. We refer the reader to [38] for an excellent exposure to the SPA on FGs, and to [23, 26] for its formulation on NFGs.

Let $\mathcal{G}$ be a tree NFG (i.e. an NFG whose underlying graph is a tree) with a vertex set V and an edge set E comprised entirely of internal edges, i.e. $\mathcal{G}$ has no external edges. To facilitate notation, if nodes u and v are neighbors, we use e_{uv} to denote the edge $\{u, v\}$. Note that $e_{uv} = e_{vu}$. The SPA defines a set of *messages* that are passed between adjacent nodes, where we use $\mu_{u\to v}$ to denote the message passed from node u to node v. The messages are governed by the following *update rule* (Fig. 5.23):

$$\mu_{u\to v}(x_{e_{uv}}) = \sum_{x\in\mathcal{X}_{E(u)\setminus\{e_{uv}\}}} f_u(x, x_{e_{uv}}) \prod_{v'\in \mathrm{ne}(u)\setminus\{v\}} \mu_{v'\to u}(x_{e_{v'u}}),$$

where a node u sends its message to adjacent node v only if it has received the messages of all its other neighbors, and the algorithm terminates when every node has sent a message to all its neighbors. That is, the message $\mu_{u\to v}$ is the sum-of-products $\big\langle f_u, \mu_{v'\to u} : v' \in \mathrm{ne}(u)\setminus\{v\}\big\rangle$, and the complexity of such computation is $(\deg(u)-1)|\mathcal{X}|^{\deg(u)-1}$. In the special case when f_u is the indicator function $\delta_{\sum}$ or $\delta_{\max}$, it is not hard to see that the complexity becomes $(\deg(u)-2)|\mathcal{X}|^2$. Further, it is clear that when f_u is the equality indicator, the message sent to node v is simply the multiplication of the incoming messages from all other neighbors of u, i.e.,

$$\mu_{u\to v}(x_{e_{uv}}) = \prod_{v'\in \mathrm{ne}(u)\setminus\{v\}} \mu_{v'\to u}(x_{e_{uv}}),$$

and the complexity of computing such message is $(\deg(u)-2)|\mathcal{X}|$.

Note that upon termination, the SPA would have computed $2|E|$ messages with two messages per edge, and it is possible to show, due to the tree structure, that each marginal $Z_{\mathcal{G}}(x_e)$ is given as the product of the two messages carried by the edge e. That is, for any $e_{uv} \in E$, we have

$$Z_{\mathcal{G}}(x_{e_{uv}}) = \mu_{u\to v}(x_{e_{uv}})\mu_{v\to u}(x_{e_{uv}}).$$

We remark that although the SPA was formulated on NFGs with no external edges, this does not present a serious limitation, as in many applications one converts the external edges, if present, into regular ones by gluing the constant-one or the evaluation indicators. (For each external edge the choice of the indicator function depends on interest and whether such edge is observed as "evidence" or not, cf. Section 5.6.) However, if one insists on NFGs with external edges, then the SPA algorithm still works for such NFGs, and it is possible to show that in this case if L is the set of external edges, then for any internal edge $e_{uv} \in E$, the product $\mu_{u\to v}\mu_{v\to u}$ is equal to $Z_{\mathcal{G}}(x_{e_{uv}}, x_L)$ defined as $Z_{\mathcal{G}}(x_{e_{uv}}, x_L) := \sum_{x_{E\setminus(L\cup\{e_{uv}\})}} \prod_{v\in V} f_v(x_{E(v)})$, and hence, the exterior function is given as $Z_{\mathcal{G}}(x_L) = \sum_{x_{e_{uv}}} Z_{\mathcal{G}}(x_{e_{uv}}, x_L)$. However, in this case, the SPA might be expensive to perform since the size of each message increases every time it is passed by a node with a dangling edge, as the message accumulates the variable of such edge as an argument.

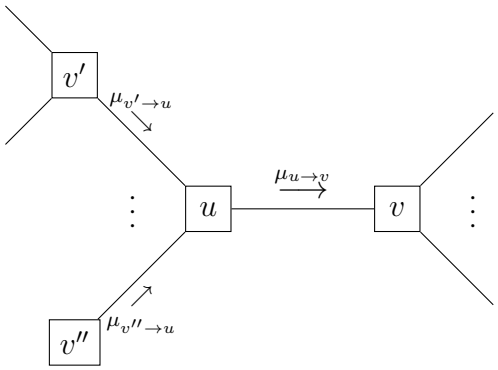


Figure 5.23: SPA update rule, where the message sent from node u to its neighbor v is computed using all the messages arriving to u from all its other neighbors, and its local function f_u, namely, $\mu_{u\to v} = \langle f_u, \mu_{v'\to u}, \ldots, \mu_{v''\to u}\rangle$.

Below, we give a simple example illustrating the SPA.

Example 8 (SPA example) *Given the NFG in Fig. 5.24, we have*

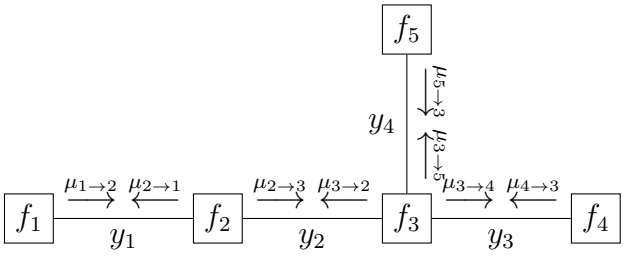


Figure 5.24: SPA example.

$$\mu_{1\to2}(y_1) = f_1(y_1), \qquad \mu_{4\to3}(y_3) = f_4(y_3),$$

$$\mu_{2\to3}(y_2) = \sum_{y_1} f_2(y_1, y_2)\mu_{1\to2}(y_1), \qquad \mu_{3\to5}(y_4) = \sum_{y_2,y_3} f_3(y_2, y_3, y_4)\mu_{2\to3}(y_2)\mu_{4\to3}(y_3),$$

$$\mu_{5\to3}(y_4) = f_5(y_4), \qquad \mu_{3\to2}(y_2) = \sum_{y_3,y_4} f_3(y_2, y_3, y_4)\mu_{4\to3}(y_3)\mu_{5\to3}(y_4),$$

$$\mu_{3\to4}(y_3) = \sum_{y_2,y_4} f_3(y_2, y_3, y_4)\mu_{2\to3}(y_2)\mu_{5\to3}(y_4), \quad \mu_{2\to1}(y_1) = \sum_{y_2} f_2(y_1, y_2)\mu_{3\to2}(y_2),$$

and it is clear by direct substitution that, for instance,

$$Z_{\mathcal{G}}(y_2) = \mu_{2\to3}(y_2)\mu_{3\to2}(y_2).$$

Another example is provided below in relation to the difference transform and the "derivative sum-product algorithm" of Huang and Frey [31].

Example 9 *Let $\mathcal{G}$ be as in Fig. 5.25. The SPA computes the following messages:*

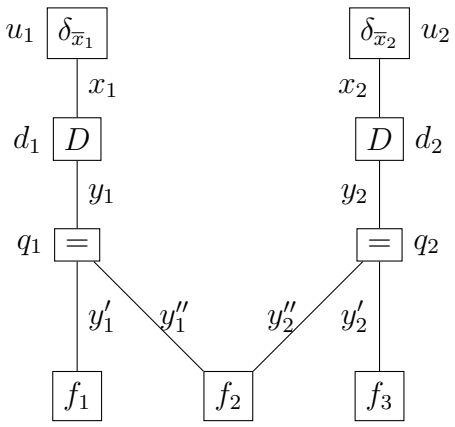


Figure 5.25: Example 9.

$$\mu_{u_1\to d_1}(x_1) = \delta_{\overline{x}_1}(x_1); \qquad \mu_{u_2\to d_2}(x_2) = \delta_{\overline{x}_2}(x_2);$$
$$\mu_{d_1\to q_1}(y_1) = \textstyle\sum_{x_1} D(x_1,y_1)\mu_{u_1\to d_1}(x_1); \qquad \mu_{d_2\to q_2}(y_2) = \textstyle\sum_{x_2} D(x_2,y_2)\mu_{u_2\to d_2}(x_2);$$
$$\mu_{f_1\to q_1}(y_1') = f_1(y_1'); \qquad \mu_{q_2\to f_3}(y_2') = \mu_{f_2\to q_2}(y_2')\mu_{d_2\to q_2}(y_2');$$
$$\mu_{q_1\to f_2}(y_1'') = \mu_{d_1\to q_1}(y_1'')\mu_{f_1\to q_1}(y_1''); \qquad \mu_{q_2\to f_2}(y_2'') = \mu_{d_2\to q_2}(y_2'')\mu_{f_3\to q_2}(y_2'');$$
$$\mu_{f_2\to q_2}(y_2'') = \textstyle\sum_{y_1''} f_2(y_1'',y_2'')\mu_{q_1\to f_2}(y_1''); \qquad \mu_{f_2\to q_1}(y_1'') = \textstyle\sum_{y_2''} f_2(y_1'',y_2'')\mu_{q_2\to f_2}(y_2'');$$
$$\mu_{f_3\to q_2}(y_2') = f_3(y_2'); \qquad \mu_{q_1\to f_1}(y_1') = \mu_{f_2\to q_1}(y_1')\mu_{d_1\to q_1}(y_1');$$
$$\mu_{q_2\to d_2}(y_2) = \mu_{f_2\to q_2}(y_2)\mu_{f_3\to q_2}(y_2); \qquad \mu_{q_1\to d_1}(y_1) = \mu_{f_1\to q_1}(y_1)\mu_{f_2\to q_1}(y_1);$$
$$\mu_{d_2\to u_2}(x_2) = \textstyle\sum_{y_2} D(x_2,y_2)\mu_{q_2\to d_2}(y_2); \qquad \mu_{d_1\to u_1}(x_1) = \textstyle\sum_{y_1} D(x_1,y_1)\mu_{q_1\to d_1}(y_1).$$

By noting that $\mu_{d_1\to q_1}(y_1) = D(\overline{x}_1, y_1)$ *and* $\mu_{d_2\to q_2}(y_2) = D(\overline{x}_2, y_2)$*, one can see by direct substitution that*

$$\mu_{d_1\to u_1}(x_1) = \sum_{y_1} D(x_1,y_1)f_1(y_1)\sum_{y_2''} D(\overline{x}_2,y_2'')f_2(y_1,y_2'')f_3(y_2''),$$

and

$$\mu_{d_2\to u_2}(x_2) = \sum_{y_2} D(x_2,y_2)f_3(y_2)\sum_{y_1''} D(\overline{x}_1,y_1'')f_2(y_1'',y_2)f_1(y_1'').$$

If we use the notation $\partial_x f(x,y) := \sum_z D(x,z)f(x,y)$ *to denote the difference transform of a function* f *(with respect to* x*), then the above two messages can equivalently be written as:*

$$\mu_{d_1\to u_1}(x_1) = \partial_{x_1}\big[f_1(x_1)\partial_{x_2}[f_2(x_1,x_2)f_3(x_2)]|_{x_2=\overline{x}_2}\big],$$

which is the difference transform of the product of all hidden functions with respect to x_1, x_2 *evaluated at* $x_2 = \overline{x}_2$*, and similarly*

$$\mu_{d_2 \to u_2}(x_2) = \partial_{x_2}\big[f_3(x_2)\partial_{x_1}[f_2(x_1, x_2)f_1(x_1)]|_{x_1=\overline{x}_1}\big]$$

is the difference transform evaluated at $x_1 = \overline{x}_1$*.*

Let $\mathcal{G}$ be a tree NFG that is also a constrained NFG with a set I of interface nodes consisting of equality indicators. Let $\mathcal{G}'$ be the NFG obtained from $\mathcal{G}$ by gluing on each dangling edge a difference function, and $\mathcal{G}''$ be the NFG resulting from $\mathcal{G}'$ by gluing an evaluation function on the other end of each such difference function, cf. Fig. 5.25. Performing the SPA on $\mathcal{G}''$, it is clear that the discussion in the previous example extends to any such $\mathcal{G}''$. That is, the message $\mu_{d_i \to u_i}(x_i)$ is the difference transform of the product of the hidden functions with respect to x_I evaluated at $\overline{x}_{I\setminus\{i\}}$, where d_i is the difference node adjacent to interface node i and u_i is the evaluation node adjacent to d_i. This is the "derivative-sum-product" algorithm of [31] in the case of finite alphabets.

We close with two remarks: First, we emphasize that in performing the SPA over $\mathcal{G}''$, one may utilize the fast difference algorithm in computing the messages emitted by the difference nodes, making the complexity of computing such messages linear in the alphabet size. Second, as one may expect, marginalization by summation on $\mathcal{G}'$ is marginalization by evaluation on $\mathcal{G}$, and the difference function guarantees the conversion between such notions of marginalization, as demonstrated in the example below.

Example 10 *Assume the function of node* u_1 *in the NFG in Fig. 5.25 is replaced with the constant-one indicator (this accounts to marginalizing* x_1 *by summing it out), and assume all variables take their values from a set* $\mathcal{X}$*. Then we have* $\mu_{u_1 \to d_1}(x_1) = 1$*, and from the definition of the difference function, it is clear that* $\mu_{d_1 \to q_1}(y_1) = \delta_{|\mathcal{X}|}(y_1)$*. By direct substitution, it follows that*

$$\mu_{d_2 \to u_2}(x_2) = \sum_{y_2} D(x_2, y_2) f_1(|\mathcal{X}|) f_2(|\mathcal{X}|, y_2) f_3(y_2).$$

Or equivalently,

$$\mu_{d_2 \to u_2}(x_2) = \partial_{x_2}\big[f_1(|\mathcal{X}|) f_2(|\mathcal{X}|, x_2) f_3(x_2)\big]$$

That is, the message $\mu_{d_2 \to u_2}$ is the difference transform with respect to x_2 of the multiplication of the hidden functions, with x_1 marginalized by evaluation at $|\mathcal{X}|$. One may verify that $\mu_{d_1 \to u_1}$ remains unchanged.

5.5.3 Indirect evaluation of the exterior function

Below, we discuss an indirect method for finding the exterior function of a bipartite NFG $\mathcal{G}$, where a holographic transformation, a one that preserves the exterior function, is applied to the NFG *a priori* in hope of facilitating the computation. The idea is to perform a transformation, if it exists, that replaces each function f_i with an equality indicator, and hence benefit from the low computational complexity of such nodes in the elimination or the SPA. Of course one might not be able to find a holographic transformation that converts each function f_i into an equality indicator, in which case, one may settle with one that produces such effect for a subset $I' \subseteq I$. For this approach to work, it is necessary that: 1) There exists a set of transformers $\{g_e : e \in E\}$ such that $\langle f_i, g_e : e \in E(i) \rangle = \delta_=$, for all $i \in I'$ for some non-empty I', and 2) There exists efficient algorithms for computing the transformations induced by g_e. The following example demonstrates this approach.

Example 11 *Let $\mathcal{X}$ be a finite ordered set and consider the NFG in Fig. 5.26 (a) where each edge variable assumes its values from the partially-ordered set $\mathcal{X}^n$ for some integer n. Directly computing the exterior function requires $|\mathcal{X}|^{2n}$ operations. However, the exterior function may be computed using a number of operations of order $n|\mathcal{X}|^n$ by first computing the fast cumulus transforms of f_1 and f_2, multiply the resulting two functions, and then invert the result using the fast difference transform, Fig. (c). The exterior function is invariant under such procedure since it simply accounts to the holographic transformation in Fig. (b), which is equivalent to the NFG in (a) by the GHT and to the one in (c) by Lemma 10.*

The discussion above parallels the well-known fast Fourier transform approach to finding the convolution of functions. In this case, the fast Fourier transform is used to reduce the complexity of computing the convolution of two functions defined on $\mathcal{X}^n$ (assuming

$\mathcal{X}$ is a finite abelian group) from order $|\mathcal{X}|^{2n}$ to $n\log(|\mathcal{X}|)|\mathcal{X}|^n$ by first computing the fast Fourier transforms of f_1 and f_2, multiply the resulting two functions, and then invert the result using the fast Fourier inverse transform. This is justified by the relation between the sum and parity indicators, and the duality of the equality and parity indicators under the Fourier transform, Figs 5.26 (d)–(f).

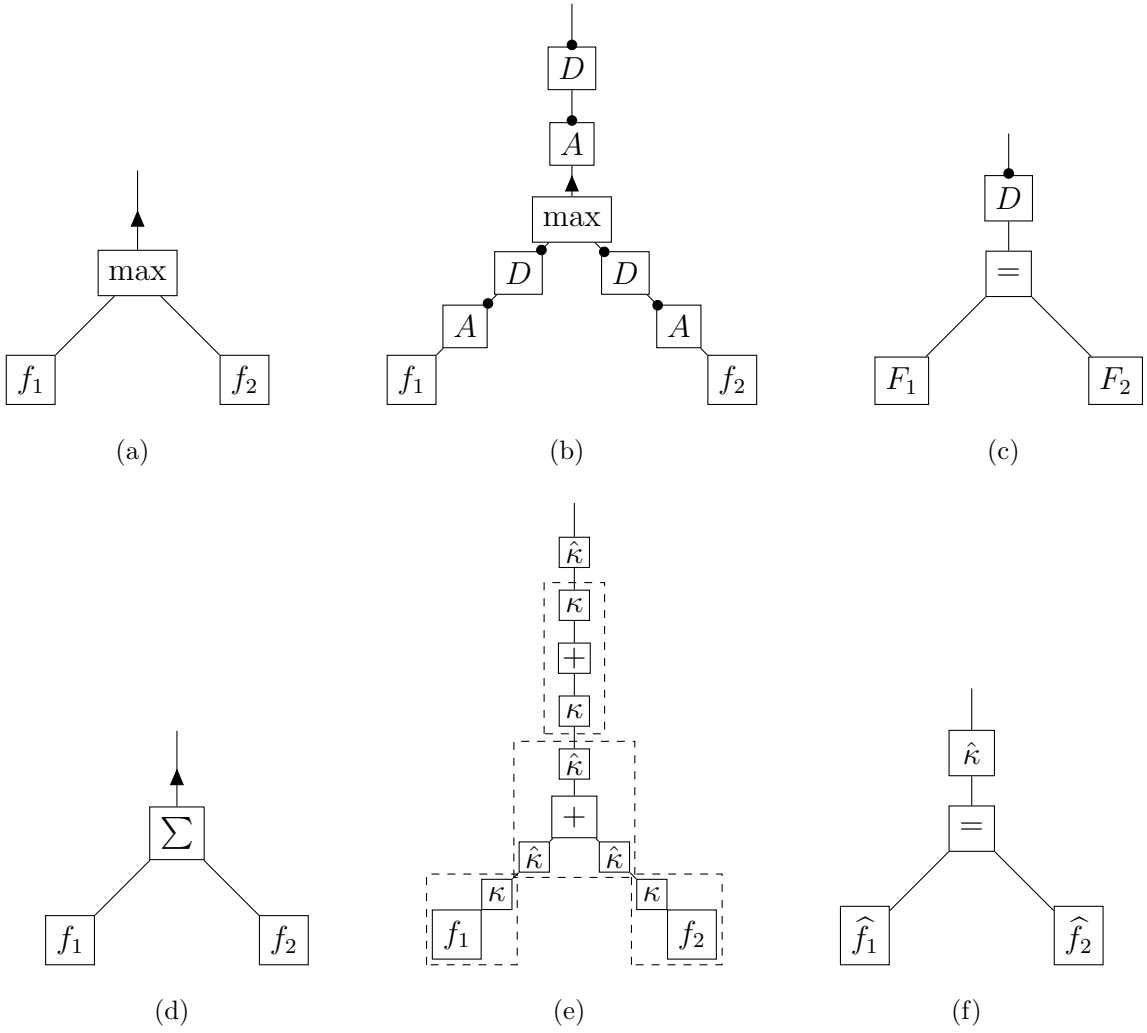


Figure 5.26: Indirect computation of the exterior function, where F_j, and $\widehat{f}_j$ are the cumulus and Fourier transforms of f_j, respectively.

5.6 The inference problem

Let an NFG $\mathcal{G}(V, E, f_V)$ represent a set of RVs X_L. (That is, the exterior function of $\mathcal{G}$ is the probability distribution p_{X_L}.) The inference problem is to marginalize a set $M \subseteq L$ of the RVs and to evaluate p_{X_L} at some observed values (evidence) of RVs $N \subseteq L$, where M and N are disjoint. That is, the problem is to find

$$p_R(x_R, \overline{x}_N) = \sum_{x_M} p_{X_L}(x_R, x_M, x_N)|_{x_N=\overline{x}_N},$$

where $R = L \backslash (M \cup N)$. We remark that, in general, p_R is not a probability distribution over the RVs X_R. It is rather an *up to scale* distribution over X_R, namely, it is the conditional distribution $p_{X_R|x_N}(x_R|x_N = \overline{x}_N)$ up to the scaling constant $p_{X_N}(\overline{x}_N)$. Evidently, the complexity of the inference problem depends primarily on the factorization structure of p_{X_L}, which is reflected by the graphical structure of the NFG. In order to perform the desired inference, we define $\mathcal{G}^*$ as the NFG whose exterior function is the desired function p_R. That is, we define $\mathcal{G}^*$ as the NFG obtained from $\mathcal{G}$ by: 1) Converting each dangling edge $e \in M$ into a regular edge by gluing a new vertex u_e to e, where u_e is associated the constant-one function. 2) Convert each dangling edge $e \in N$ into a regular edge by gluing a new vertex v_e to e, where v_e is associated the evaluation indicator $\delta_{\overline{x}_e}$. An example is shown in Fig. 5.27 where the original NFG is as in (a). Assuming we are interested in $\sum_{x_3} p_{X_1X_2X_3}(x_1, x_2, x_3)|_{x_2=\overline{x}_2}$, then $\mathcal{G}^*$ is as in (b).

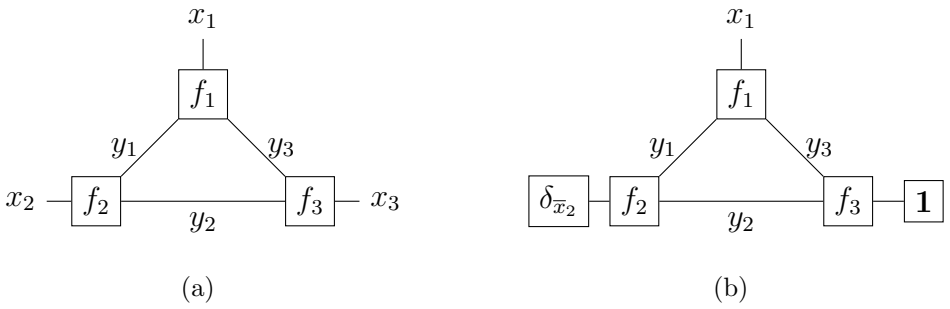


Figure 5.27: Inference: (a) An example NFG $\mathcal{G}$ representing $p_{X_1X_2X_3}(x_1, x_2, x_3)$, (b) the resulting $\mathcal{G}^*$ assuming we are interested in $\sum_{x_3} p_{X_1X_2X_3}(x_1, x_2, x_3)|_{x_2=\overline{x}_2}$.

Clearly the inference problem is encoded in $\mathcal{G}^*$, and hence reduces to computing the

exterior function of $\mathcal{G}^*$, which can be performed by invoking the elimination algorithm on $\mathcal{G}^*$. From this equivalence between inference and the computation of the exterior function, one can always assume that the given NFG represents the desired computation, i.e., one can assume that the NFG is already reduced to the desired inference $(.)^*$ form.

We remark that in a constrained model, by Proposition 3, we may assume that each interface node is an equality indicator. Hence, for each evaluated RV, i.e., for each $i \in N \subseteq I$, we may 1) for each neighbor $j \in \text{ne}(i)$ of i, connected to i by edge e, replace f_j with $f_j(x_{E(j)})|_{x_e=\overline{x}_e}$ and delete e, and 2) delete node i. Hence, the inference problem over constrained models is simply a marginalization one, and one may always assume N is empty.

On the other hand, for a conditional function of x given y, we have $\sum_x f(x, y)$ is a constant c independent of y. Hence, in a generative model $\mathcal{G}$ with vertex set $I \cup J$, for each marginalized RV, i.e., for each $i \in M \subseteq I$, we may 1) absorb the constant c_i into one of the neighbors of i by replacing f_j with $c_i f_j$ for some $j \in \text{ne}(i)$, 2) for each neighbor $j \in \text{ne}(i)$ of i, connected to i by edge e, replace f_j with $\sum_{x_e} f_j(x_{E(j)})$ and delete e, and 3) delete node i. Hence, the inference problem over generative models is simply an evaluation one, and one may always assume M is empty.

Chapter 6

Translation Association Schemes

This chapter focuses on a subclass of association schemes [20, 21], namely, the class of translation association schemes, and presents a relatively simple approach to understanding them. In particular, we choose to formulate such schemes using the notion of convolution from linear-system theory and use normal factor graphs in establishing some of the proofs, which appears to be appealing. Furthermore, we discuss two important extensions (ordered and unordered) of translation schemes, which are by themselves translation schemes. Duality results concerning translation schemes and their extensions are also presented. The most basic form of linear programming (LP) bound for translation scheme is derived and we show that when the translation scheme of interest is specified as the ordered extension of a one-class translation scheme, the LP bound reduces to the LP bound of Chan et al. [13] based on the notion of "support enumerator".

6.1 Preliminaries

We denote sets by calligraphic letters, e.g., $\mathcal{X}, \mathcal{Y}, \ldots$. To simplify notation, for any subset $\mathcal{Y} \subseteq \mathcal{X}$, we use Y (the corresponding non-calligraphic letter) to denote the function $[x \in \mathcal{Y}]$, where the domain of the function (in this case $\mathcal{X}$) should be clear from the context. (Recall that for any Boolean proposition P, notation $[P]$ evaluates to 1 if P is "true", and evaluates to 0 otherwise.) We say Y is the *membership* function of $\mathcal{Y}$, and

$\mathcal{Y}$ is the *support* of Y.

The notions of convolution and the Fourier transform play fundamental roles in the theory of linear time-invariant (LTI) systems. In fact, LTI system theory well generalizes to the case where "time" takes values in abelian groups. As usual, we use "$*$" to denote the convolution. However, for any function f on an abelian group G and any function g on the character group $\widehat{G}$, we find it more convenient in this chapter to use $\hat{f}$ and $\check{g}$ to denote the Fourier and Fourier inverse transforms of f and g, respectively. (Note also that in this chapter we use $\widehat{G}$ to denote the character group instead of $G^{\wedge}$.)

Generalizing a well-known result in system theory, it holds that $\widehat{f * g} = \hat{f} \cdot \hat{g}$ for functions f and g on a finite abelian group, where we have used "$\cdot$" to denote element-wise product, or simply, multiplication.

For any function f, let f^- denote the function f with the variable's sign inverted, i.e., $f^-(x) = f(-x)$. In classical system theory, $f * f^-$ is then the *autocorrelation function* of f, and it is well known that the Fourier transform of an autocorrelation function is an energy spectral density, and so is always nonnegative. Such result also holds in the setting below.

Lemma 12 *For any real function f on an abelian group, $\widehat{f * f^-}$ is nonnegative.*

Proof: Let f be a real function on G, then

$$\begin{aligned}
\widehat{f * f^-}(\hat{x}) &= \sum_{x \in G} (f * f^-)(x) \hat{x}(x) \\
&= \sum_{x \in G} \sum_{y \in G} f(y) f^-(x - y) \hat{x}(x) \\
&= \sum_{x \in G} \sum_{y \in G} f(y) f(y - x) \hat{x}(x) \\
&= \sum_{y \in G} f(y) \sum_{x \in G} f(x) \hat{x}(y - x) \\
&= \sum_{y \in G} f(y) \hat{x}(y) \sum_{x \in G} f(x) \overline{\hat{x}(x)} \\
&= \Big| \sum_{x \in G} f(x) \hat{x}(x) \Big|^2,
\end{aligned}$$

where the last equality follows since f is real. ■

As we will see later, this lemma turns out to be fundamental in establishing LP bounds with respect to translation schemes.

6.2 Translation schemes

Let G be a finite abelian group, $\mathcal{D} = \{0, 1, \ldots, d\}$ for some positive integer $d \leq |G|$, and $\mathfrak{L} \to \mathcal{D}$ be a surjective map from G to $\mathcal{D}$. (Such assumptions are made throughout the entire chapter without explicitly stating them in every occasion.) At many places, we use $L_i(x)$ to denote the indicator function $[\mathfrak{L}(x) = i]$ for all $x \in G$ and $i \in \mathcal{D}$. While $L_i(x)$ may be understood as a bivariate function of x and i, it is also useful to understand it as a univariate function that is indexed by i, and so view $\{L_i : i \in \mathcal{D}\}$ as a family of functions. Both views are useful and are used in this work. Note that the support $\mathcal{L}_i$ of L_i is the preimage of i under $\mathfrak{L}$, i.e., $\mathcal{L}_i = \{x \in G : \mathfrak{L}(x) = i\}$, and that $\{\mathcal{L}_i : i \in \mathcal{D}\}$ is a partition of G. Subsequently, the index i of class $\mathcal{L}_i$ is referred to as the *type* of class $\mathcal{L}_i$. Further, the type of the class to which an element $x \in G$ belongs is also said to be the type of x. We now introduce the notion of translation association scheme [7, 11, 21], or translation scheme in short, which we choose to formulate using the notion of convolution.

Definition 1 (Translation Scheme) *Let G be a finite abelian group, $\mathcal{D} = \{0, \ldots, d\}$ with $d \leq |G|$, and $\mathfrak{L} : G \to \mathcal{D}$ be surjective. Then $(G, \mathfrak{L})$ is said to be a d-class translation scheme if:*

i. $\mathcal{L}_0 = \{0\}$, where 0 is the identity element of G.

ii. There exist integers p_{ij}^k, called the p-numbers, such that for any $i, j \in \mathcal{D}$

$$L_i * L_j = \sum_{k=0}^{d} p_{ij}^k L_k.$$

We make few remarks on the definition above. Since convolution is commutative, it is clear that $p_{ij}^k = p_{ji}^k$ for all $i, j, k \in \mathcal{D}$. In fact, the p-numbers have a combinatorial meaning. Namely, p_{ij}^k is the number of ways an element $x \in G$ of type k can be written as the sum of two elements of types i and j. Condition (ii) states that such number

depends on x through the type of x and not the particular choice of x. That is, condition (ii) above may be replaced with:

ii.′ For all $i, j, k \in \mathcal{D}$ and all $x \in \mathcal{L}_k$,

$$|\{(y, z) \in \mathcal{L}_i \times \mathcal{L}_j : x = y + z\}| = p_{ij}^k,$$

for some integers p_{ij}^k. Finally, it is possible to show that if $(G, \mathfrak{L})$ is a translation scheme, then for any $i \in \mathcal{D}$, it holds that $-\mathcal{L}_i = \mathcal{L}_j$ for some $j \in \mathcal{D}$, where for any set $\mathcal{X}$, the set $-\mathcal{X}$ is defined as $-\mathcal{X} = \{-x : x \in \mathcal{X}\}$. (See Appendix B.2.)

Example 12 *Let $G = \mathbb{Z}_m$, be the group of integers modulo m, and let $\mathcal{D} = \{0, \ldots, d\}$, where $d = \frac{m}{2}$ if m is even and $d = \frac{m-1}{2}$ if m is odd. For any element $x \in G$, define $|x|$ as $|x| = x$ if $x \in \mathcal{D}$ and $|x| = -x$ if $x \notin \mathcal{D}$, and let $\mathfrak{L}(x) = |x|$ for all $x \in G$. (Note that the size of any set $\mathcal{L}_i$ is either 1 or 2.) Clearly, $\mathcal{L}_0 = \{0\}$, and for all $i, j \in \mathcal{D}$, it is possible to show that*

$$L_i * L_j(x) = \begin{cases} \frac{2}{|\mathcal{L}_{|i+j|}|} L_{|i+j|} + \frac{2}{|\mathcal{L}_{|i-j|}|} L_{|i-j|}, & i \neq -i, j \neq -j, \\ L_{|i+j|}, & \text{otherwise.} \end{cases}$$

From this, we have $(G, \mathfrak{L})$ is a d-class translation scheme.

Given a translation scheme $(G, \mathfrak{L})$, let $\mathcal{A}$ be the complex span of the indicator functions $\{L_i : i \in \mathcal{D}\}$, for which we also write $\mathcal{A} = \text{span}\{L_i : i \in \mathcal{D}\}$. The fact that $\{\mathcal{L}_i : i \in \mathcal{D}\}$ is a partition (of G) implies that $\mathcal{A}$ is a vector space of dimension $d + 1$ for which $\{L_i : i \in \mathcal{D}\}$ is a basis. Since convolution is linear in both of the involved functions, condition (ii) of Definition 1 simply says that $\mathcal{A}$ is closed under convolution. Further, $\mathcal{A}$ is also closed under (function) multiplication since the basis $\{L_i : i \in \mathcal{D}\}$ consists of indicator functions of disjoint subsets of G. That is, $\mathcal{A}$ is a commutative algebra under convolution and multiplication.

Let $\hat{L}_i$ be the Fourier transform of L_i and $\hat{\mathcal{A}} = \text{span}\{\hat{L}_i : i \in \mathcal{D}\}$. Then every element in $\hat{\mathcal{A}}$ is the Fourier transform of an element in $\mathcal{A}$. It follows that $\hat{\mathcal{A}}$ is also a vector space closed under convolution and multiplication. The commutative algebra $\hat{\mathcal{A}}$ can be shown to possesses a basis which are indicator functions $\{L_i' : i \in \mathcal{D}\}$ for some

partition $\{\mathcal{L}'_i : i \in \mathcal{D}\}$ of $\widehat{G}$. (Such basis is unique subject to indexing.) This defines a map $\mathfrak{L}' : \widehat{G} \to \mathcal{D}$ such that $\mathfrak{L}'(\hat{x}) = i$ if $\hat{x} \in \mathcal{L}'_i$, and it follows that $(\widehat{G}, \mathfrak{L}')$ is also a d-class translation scheme [11, 20], commonly referred to as the *dual* scheme of $(G, \mathfrak{L})$. (See Appendix B.1.)

Viewed from the perspectives of the translation scheme $(G, \mathfrak{L})$ and its dual $(\widehat{G}, \mathfrak{L}')$ respectively, $\mathcal{A}$ has two canonical bases, $\{L_i : i \in \mathcal{D}\}$ and $\{\check{L}'_i : i \in \mathcal{D}\}$, where $\check{L}'_i$ denotes the Fourier inverse of L'_i. Expressing each element of the basis $\{\check{L}'_i : i \in \mathcal{D}\}$ in terms of the basis $\{L_i : i \in \mathcal{D}\}$, we have

$$\check{L}'_i = \sum_{j \in \mathcal{D}} L_j Q(j, i), \tag{6.1}$$

for an invertible and uniquely determined $|\mathcal{D}| \times |\mathcal{D}|$ matrix Q.

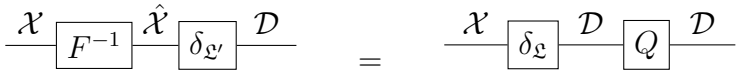


Figure 6.1: A graphical illustration of (6.1), where F^{-1} is the Fourier inverse kernel, $\delta_{\mathfrak{L}}(x, i) := [\mathfrak{L}(x) = i]$, and $\delta_{\mathfrak{L}'}(\hat{x}, j) := [\mathfrak{L}'(\hat{x}) = j]$ for all $x \in G$, $\hat{x} \in \widehat{G}$ and all $i, j \in \mathcal{D}$.

Likewise, $\hat{\mathcal{A}}$ has two canonical bases, $\{\hat{L}_i : i \in \mathcal{D}\}$ and $\{L'_i : i \in \mathcal{D}\}$, which are related by

$$\hat{L}_i = \sum_{j \in \mathcal{D}} L'_j Q^{-1}(j, i), \tag{6.2}$$

Equation (6.1) states that the Fourier inverse $\check{L}'_i(x)$ depends on x only through its type, rather than the particular choice of x. That is, assuming $x \in \mathcal{L}_j$, then

$$\sum_{\hat{x} \in \mathcal{L}'_i} \hat{x}(x) = |G| \overline{Q(j, i)}, \tag{6.3}$$

where $\overline{(\cdot)}$ denotes the complex conjugate. Similarly, (6.2) says that the Fourier transform $\hat{L}_i(\hat{x})$ depends only on the type of $\hat{x}$. That is, assuming $\hat{x} \in \mathcal{L}'_j$, then

$$\sum_{x \in \mathcal{L}_i} \hat{x}(x) = Q^{-1}(j, i), \tag{6.4}$$

In [69], two partitions of G and $\widehat{G}$ satisfying (6.3) and (6.4) are called *Fourier-invariant*.

The matrix Q defines a linear map on the space of functions on $\mathcal{D}$. Namely, for any function g on $\mathcal{D}$ its *Q-transform* is defined as

$$\widetilde{g}(j) = \sum_{i \in \mathcal{D}} g(i)Q(i,j), \tag{6.5}$$

for all $j \in \mathcal{D}$.

Example 13 (One-class translation scheme) *Let G be a finite abelian group, and for all $x \in G$, let $\mathfrak{L}(x) = 0$ if $x = 0$ and $\mathfrak{L}(x) = 1$ otherwise. Then it is easy to verify that $(G, \mathfrak{L})$ is a one-class translation scheme. We have,*

$$L_0(x) = \begin{cases} 1, & x = 0, \\ 0, & \text{otherwise}, \end{cases} \quad \text{and } L_1(x) = \begin{cases} 0, & x = 0, \\ 1, & \text{otherwise}. \end{cases}$$

Hence, $\hat{L}_0(\hat{x}) = 1$ for all $\hat{x} \in \hat{G}$, and

$$\hat{L}_1(\hat{x}) = \begin{cases} |G| - 1, & \hat{x} = 0, \\ -1, & \text{otherwise}. \end{cases}$$

From this it follows that

$$L'_0(\hat{x}) = \begin{cases} 1, & \hat{x} = 0, \\ 0, & \text{otherwise}, \end{cases} \quad \text{and } L'_1(\hat{x}) = \begin{cases} 0, & \hat{x} = 0, \\ 1, & \text{otherwise}, \end{cases}$$

and so,

$$\hat{L}_0 = L'_0 + L'_1 \text{ and } \hat{L}_1 = (|G| - 1)L'_0 - L'_1.$$

Comparing this to (6.2), it follows that

$$Q^{-1}(i,j) = \begin{cases} |G| - 1, & i = 0, j = 1, \\ -1, & i = 1, j = 1, \\ 1, & \text{otherwise}, \end{cases}$$

and it is easy to verify that $Q = \frac{1}{|G|}Q^{-1}$. This example is of particular interest since its "unordered extension" (Section 6.3.2) is the classical Hamming scheme from coding theory.

6.3 Extension translation schemes

In this section we focus on two particular translation schemes, the "ordered extension" and "unordered extension" translation schemes. Here the word "extension" is in the sense of constructing an object from "smaller" ones, and not in the sense of generalizing.

6.3.1 Ordered extension

For any positive integer n, let $\mathcal{N} = \{1, \ldots, n\}$. For all $k \in \mathcal{N}$, let $(G_k, \mathfrak{L}_k)$ be a d_k-class translation, and let $\mathcal{D}_k = \{0, \ldots, d_k\}$. Define

$$\mathfrak{L}_O(x) = (\mathfrak{L}_1(x_1), \ldots, \mathfrak{L}_n(x_n)), \tag{6.6}$$

for all $x = (x_1, \ldots, x_n) \in G_1 \times \cdots \times G_n$. Then it is possible to show that $(G_1 \times \cdots \times G_n, \mathfrak{L}_O)$ is a d-class translation scheme with $d = \prod_{k \in \mathcal{N}} d_k - 1$. We refer to this translation scheme as the *ordered extension translation scheme*, or simply the *ordered extension scheme*, of the *base* translation schemes $(G_k, \mathfrak{L}_k)$, $k \in \mathcal{N}$.

Further, one can show that the dual of the ordered extension scheme is the ordered extension of the dual base translation schemes. That is, for the dual ordered extension scheme $(\widehat{G}_1, \ldots, \widehat{G}_n, \mathfrak{L}'_O)$, we have for all $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_n) \in \widehat{G}_1 \times \cdots \times \widehat{G}_n$,

$$\mathfrak{L}'_O(\hat{x}) = (\mathfrak{L}'_1(\hat{x}_1), \ldots, \mathfrak{L}'_n(\hat{x}_n)). \tag{6.7}$$

From this, the result below follows.

Corollary 1 *Let Q_O be the Q-transform kernel induced by the ordered extension scheme. Then,*

$$Q_O(i, j) = \prod_{k \in \mathcal{N}} Q_k(i_k, j_k),$$

for all $i = (i_1, \ldots, i_n), j = (j_1, \ldots, j_n) \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$, where Q_k is the Q-transform kernel induced by the translation scheme $(G_k, \mathfrak{L}_k)$ for all $k \in \mathcal{N}$.

6.3.2 Unordered extension

Let $(G, \mathfrak{L})$ be a d-class translation scheme, and define the map $U : \mathcal{D}^n \to \mathcal{N}^d$ as $U(i_1, \ldots, i_n) = (\alpha_1, \ldots, \alpha_d)$ such that $\alpha_r = |\{j \in \mathcal{N} : i_j = r\}|$ for all $r \in \mathcal{D} \backslash \{0\}$. In

other words, α_r counts the number of components of the vector $(i_1, \ldots, i_n)$ that are equal to r. Let $\mathcal{D}_U = \{U(i) : i \in \mathcal{D}^n\}$ be the image of U, and note that $\sum_{r \in \mathcal{D}\setminus\{0\}} \alpha_r \leq n$ for all $(\alpha_1, \ldots, \alpha_n) \in \mathcal{D}_U$. No attention was made to the number of components of type 0. (Such number can easily be determined as $n - \sum_{r \in \mathcal{D}\setminus\{0\}} \alpha_r$ for all $\alpha \in \mathcal{D}_U$.) In the special case when $(G, \mathfrak{L})$ is the one-class translation scheme, one can verify that $U(i_1, \ldots, i_n) = i_1 + \cdots + i_n$, and that $\mathcal{D}_U = \mathcal{N}$. Now, define

$$\mathfrak{L}_U(x) = U(\mathfrak{L}(x_1), \ldots, \mathfrak{L}(x_n)), \tag{6.8}$$

then it is possible to show that $(G^n, \mathfrak{L}_U)$ is a d_U-translation scheme with $d_U = \binom{n+d}{d} - 1$ [20,29]. The translation scheme $(G^n, \mathfrak{L}_U)$ is referred to as the *unordered extension scheme* of the *base* translation scheme $(G, \mathfrak{L})$. Note that $\mathfrak{L}_U$ is the function composition $U \circ \mathfrak{L}_O$, where $\mathfrak{L}_O$ is as in the previous section. That is, the unordered extension is obtained by considering all the elements of G^n whose types in the ordered extension map under U to the same element to be of the same type. In the special case where the base scheme is the one-class translation scheme, the unordered extension scheme reduces to the classical *Hamming scheme*. On the other hand, we refer to the unordered extension of the translation scheme in Example 12 as the *Lee scheme*.

Before we proceed, we make the following remark. Let f be a bivariate function on $\mathcal{D} \times \mathcal{D}$, and for all $\alpha \in \mathcal{D}_U$ and $(j_1, \ldots, j_n) \in \mathcal{D}^n$, let

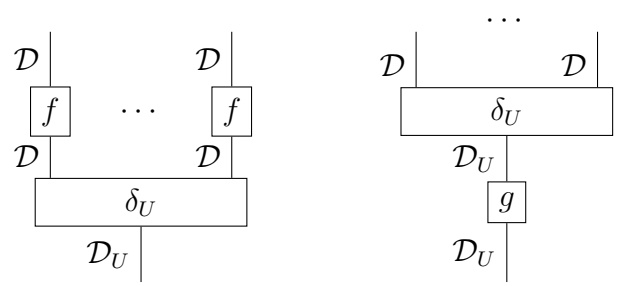
$$G(\alpha, j_1, \ldots, j_n) = \sum_{(i_1, \ldots, i_n) \in \mathcal{D}^n} f(i_1, j_1) \cdots f(i_n, j_n)[\alpha = U(i_1, \ldots, i_n)].$$

Then, since $U(i_1, \ldots, i_n) = U(i_{\sigma(1)}, \ldots, i_{\sigma(n)})$ for any permutation σ on $\mathcal{N}$, it follows that $G(\alpha, j_1, \ldots, j_n) = G(\alpha, j_{\sigma(1)}, \ldots, j_{\sigma(n)})$, and so G depends on $(j_1, \ldots, j_n)$ only through $U(j_1, \ldots, j_n)$ and not the particular choice of $(j_1, \ldots, j_n)$. That is, for some bivariate function g on $\mathcal{D}_U \times \mathcal{D}_U$, we have

$$G(\alpha, j_1, \ldots, j_n) = g(\alpha, \beta), \text{ if } U(j_1, \ldots, j_n) = \beta. \tag{6.9}$$

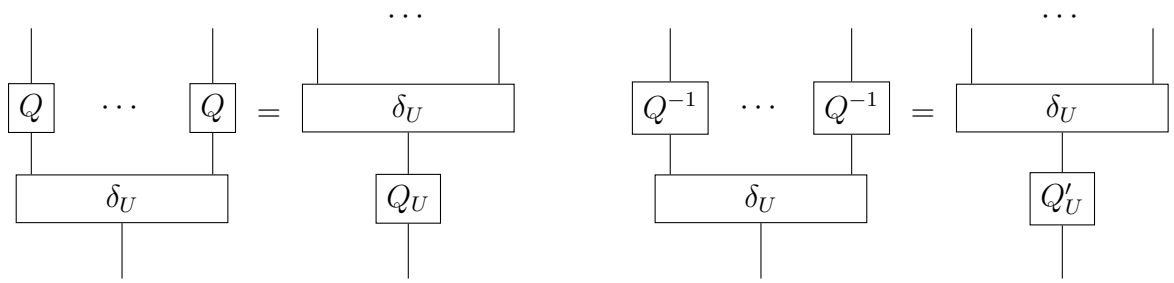
This is illustrated in Fig. 6.2.

Lemma 13 *Let Q_U and Q'_U be two $|\mathcal{D}_U| \times |\mathcal{D}_U|$ matrices such that the equalities in Fig. 6.3 (a) and (b) hold, then $Q'_U = Q_U^{-1}$.*



(a) (b)

Figure 6.2: For any function f on $\mathcal{D} \times \mathcal{D}$, the two NFGs are equal for some function g on $\mathcal{D}_U \times \mathcal{D}_U$.



(a) (b)

Figure 6.3: Lemma 13, where $\delta_U(\alpha, i) := [\alpha = U(i)]$ for all $\alpha \in \mathcal{D}_U$ and all $i \in \mathcal{D}^n$.

Proof: The proof is due to Fig. 6.4. ■

Now consider the unordered extension of the dual scheme $(\mathfrak{L}', \widehat{G})$, represented by the leftmost NFG in Fig. 6.5. By Lemma 13, Q_U is an invertible $|\mathcal{D}_U| \times |\mathcal{D}_U|$ matrix, and from (6.1) it follows that Q_U is the Q-matrix of the unordered extension and the rightmost NFG in Fig 6.5 represents the dual of the unordered extension of $(\mathfrak{L}, G)$. Hence, similar to the ordered extension scheme, we have the dual of the unordered extension scheme is the unordered extension of the dual base translation scheme. That is, for the dual unordered extension scheme $(\widehat{G}^n, \mathfrak{L}'_U)$, we have for all $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_n) \in \widehat{G}^n$,

$$\mathfrak{L}'_U(\hat{x}) = U(\mathfrak{L}'(\hat{x}_1), \ldots, \mathfrak{L}'(x_n)). \tag{6.10}$$

From this, the result below follows.

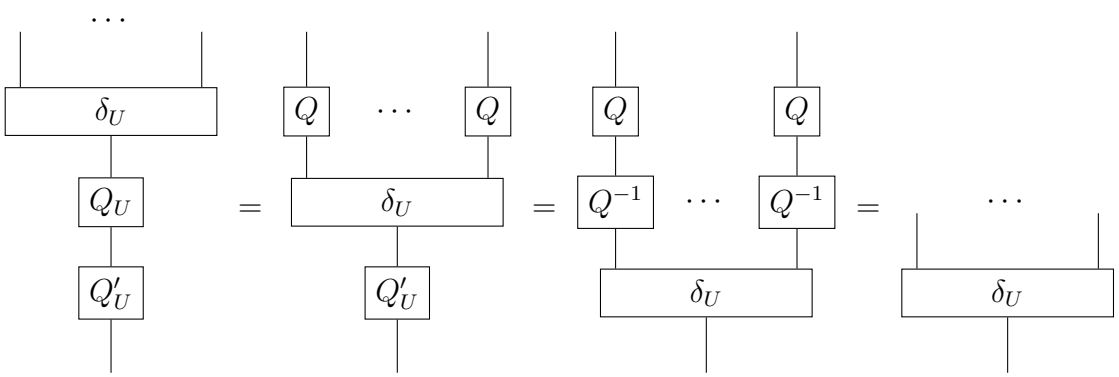


Figure 6.4: Proof of Lemma 13.

Corollary 2 *Let Q and Q_U be the Q-transform kernels induced by the base and its unordered extension translation schemes, respectively. Then, for all $\alpha \in \mathcal{D}_U$ and all $i = (i_1, \ldots, i_n) \in \mathcal{D}^n$,*

$$\sum_{\beta \in \mathcal{D}_U} Q_U(\alpha, \beta)[U(i) = \beta] = \sum_{j \in \mathcal{D}^n} \prod_{k \in \mathcal{N}} Q(j_k, i_k)[U(j) = \alpha],$$

where $j = (j_1, \ldots, j_n)$.

6.4 Type distribution, difference type distribution, and LP bound

Given a translation scheme $(G, \mathfrak{L})$, a *code* is a subset $\mathcal{Y} \subseteq G$, which in general is not required to be a group. The indicator function of a code $\mathcal{Y}$ is denoted Y.

6.4.1 Type distribution

Let $(G, \mathfrak{L})$ be a d-class translation scheme. For any function f on G, define the *type distribution function* of f (with respect to the translation scheme) as

$$T_f(i) = \sum_{x \in G} f(x) L_i(x), \tag{6.11}$$

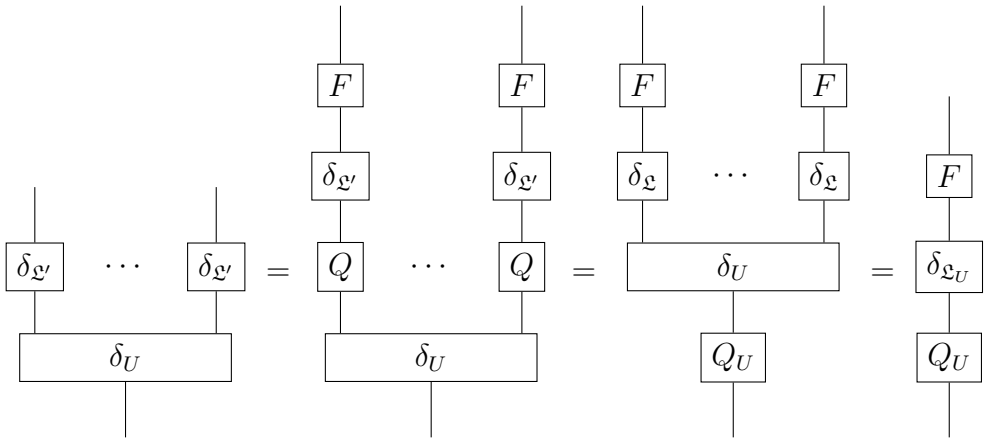


Figure 6.5: The dual of the unordered extension scheme is the unordered extension of the dual base translation scheme.

for all $i \in \mathcal{D}$. Note that when f is the indicator function Y of a code $\mathcal{Y} \subseteq G$, then

$$T_Y(i) = |\{y \in \mathcal{Y} : \mathfrak{L}(x) = i\}|,$$

which counts the number of codewords in $\mathcal{Y}$ of type i. Dually, the *dual type distribution* of f, denoted T'_f, is defined as the type distribution of $\hat{f}$ with respect to the dual scheme, i.e.,

$$T'_f(i) = \sum_{\hat{x} \in \widehat{G}} \hat{f}(\hat{x}) L'_i(\hat{x}), \tag{6.12}$$

From (6.5) and (6.1) the following theorem is obtained.

Theorem 6 *The Q-transform of the type distribution of any function f can be written as*

$$\widetilde{T}_f = \frac{1}{|G|} T'_{f^-}.$$

6.4.2 Difference type distribution

Let $(G, \mathfrak{L})$ be a d-class translation scheme. The *difference type distribution*, also called the *inner distribution* [21], of any function f on G is defined as

$$\Delta_f = T_{f*f^-}.$$

Note that when f is the indicator function Y of a code $\mathcal{Y}$, then

$$\Delta_Y(i) = |\{(x, y) \in \mathcal{Y} \times \mathcal{Y} : \mathfrak{L}(x - y) = i\}|, \tag{6.13}$$

for all $i \in \mathcal{D}$. In the special case when $\mathcal{Y}$ is a group code, then it is easy to show that $Y * Y^- = |\mathcal{Y}|Y$, and so, the type and difference type distributions are equal. (Up to a scaling factor.)

Given a code from an ordered extension scheme. Then in the special case where every base scheme is a one-class translation scheme, the difference type distribution of the code reduces to the notion of *support enumerator* of [13]. Another two useful distributions also arise when the code is from the Hamming scheme and when the code is from the Lee scheme, where the difference type distribution reduces to the *Hamming distance distribution* and the *Lee distance distribution*, respectively.

For any nonnegative function f, it is clear that $f * f^-$ is nonnegative, and so it follows immediately from (6.11) that Δ_f is nonnegative. (Recall that L_i is an indicator function.) Moreover, from Lemma 12, Theorem 6, and (6.12), the following theorem follows.

Theorem 7 *For any real function f, $\widetilde{\Delta}_f$ is nonnegative.*

Let $\mathcal{E}$ be a subset of $\mathcal{D}$. An $\mathcal{E}$-correcting code, is a code in which no two codewords y and y' are such that $y - y' \in \mathcal{L}_i$ for any $i \in \mathcal{E}$. (In other words, the set $\{\mathfrak{L}(y - y') : (y, y') \in \mathcal{Y} \times \mathcal{Y}\} \cap \mathcal{E}$ is empty.) From Theorem 7 and (6.5), the linear programming (LP) bound below follows.

Theorem 8 *Let $\mathcal{Y}$ be an $\mathcal{E}$-correcting code (from an arbitrary translation scheme), then*

$|\mathcal{Y}|^2$ *is upper bounded by the solution to the following problem:*

$$\begin{aligned}
\text{maximize :} \quad & \sum_{i\in\mathcal{D}} \Delta_Y(i), & \\
\text{subject to :} \quad & \Delta_Y(i) \geq 0, & \forall i \in \mathcal{D}, \\
& \sum_{j\in\mathcal{D}} \Delta_Y(j) Q(j,i) \geq 0, & \forall i \in \mathcal{D}, \\
& \Delta_Y(i) = 0, & \forall i \in \mathcal{E}.
\end{aligned}$$

Restricting to ordered extension schemes and evoking Corollary 1, the following result is immediate.

Corollary 3 *Let $\mathcal{Y}$ be an $\mathcal{E}$-correcting code from the ordered extension scheme $(G_1 \times \cdots \times G_n, \mathfrak{L}_O)$, then $|\mathcal{Y}|^2$ is upper bounded by the solution to the following problem:*

$$\begin{aligned}
\text{maximize :} \quad & \sum_{i\in\mathcal{D}} \Delta_Y(i), & \\
\text{subject to :} \quad & \Delta_Y(i) \geq 0, & \forall i \in \mathcal{D}, \\
& \sum_{j\in\mathcal{D}} \Delta_Y(j) \prod_{k\in\mathcal{N}} Q_k(j_k, i_k) \geq 0, & \forall i \in \mathcal{D}, \\
& \Delta_Y(i) = 0, & \forall i \in \mathcal{E},
\end{aligned}$$

where $\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$, $i = (i_1, \ldots, i_n)$, and $j = (j_1, \ldots, j_n)$.

In the special case where $\mathcal{D}_k = \{0, 1\}$ for all $k \in \mathcal{N}$. Then, as discussed in Example 13, we have up to a scaling factor, $Q_k(0,0) = Q_k(1,0) = 1$, $Q_k(0,1) = |G_k| - 1$, and $Q_k(1,1) = 1$ for all $k \in \mathcal{N}$, and this corollary reduces to the LP bound of [13].

Similarly, restricting Theorem 8 to unordered extension schemes, and using Corollary 2, one may obtain the LP bounds for codes from the Hamming and the Lee schemes. (By choosing the corresponding base scheme.)

Chapter 7

Stochastic Approximation of the Partition Function

7.1 Introduction

The estimation of partition function for statistical models is of fundamental importance in statistical physics, machine learning and information theory [49, 65]. The models we consider in this chapter are specified by a collection of random variables $\{X_i : i = 1, 2, \ldots, N\}$, for some positive integer N; each random variable X_i is assumed to take values (often called *spins*) from some finite set $\mathcal{X}$; every configuration $x \in \mathcal{X}^N$ is associated with an energy level $E(x)$, and the joint distribution of random variables $\{X_i : i = 1, 2, \ldots, N\}$ is modelled as the Boltzmann distribution

$$p_{\mathrm{B}}(x) := \frac{e^{-\beta E(x)}}{Z}, \tag{7.1}$$

for all $x \in \mathcal{X}^N$. In (7.1), $\beta := \frac{1}{kT}$ is often referred to as the "inverse temperature", where T is the temperature and k is the Boltzmann constant, and the normalizing constant $Z := \sum_{x \in \mathcal{X}^N} e^{-\beta E(x)}$ is known as the *partition function*.

Given β and the energy function $E(\cdot)$, exact computation of the partition function Z for systems involving a large number of random variables is known to be intractable, and it is precisely the intractability of this problem that roots the hardness of various

problems in coding and information theory (e.g., determining the capacity of constrained codes). Developing bounding techniques (e.g. [65]) and approximation methods [57] for estimating the partition functions is thus an active area of research.

This work is motivated by the recent empirical observation of [47] that for the two-dimensional nearest-neighbor Ising model (binary spins), the duality of normal factor graphs (NFG) [1] appears to facilitate the estimation of the partition function. In particular, they experimentally show that for large β, two stochastic estimation methods (the Ogata-Tanemura method [51] based on Gibbs sampling and a method based on uniform sampling) provide better estimation of the partition function when sampling from the dual NFG compared to sampling from the primal NFG.

In this chapter, we explain the behaviour observed in [47] and show both analytically and experimentally that such a trend extends beyond the Ising model to q-ary spins, i.e., the standard Potts model [68]. Along our development, we also provide insights on the question for what other two-dimensional nearest-neighbor models such behaviour may hold.

7.2 Preliminaries

7.2.1 Model

In Equation (7.1), we consider that each index in $\{1, 2, \ldots, N\}$ corresponds to a grid point in an $L \times L$ square lattice. We assume that the lattice is "wrapped around" in the sense that the left-most point of each row is connected to the right-most point of the same row and the top-most point of each column is connected to the bottom-most point of the same column. Let $\mathcal{A}$ denote the set of all pairs of adjacent lattice points. The energy function is assumed to take the form

$$E(x) := -\sum_{\{i,j\}\in\mathcal{A}} g_{ij}(x_i, x_j), \tag{7.2}$$

for a collection of functions $\{g_{ij} : (i,j) \in \mathcal{A}\}$. Such a model is referred to as a two-dimensional nearest-neighbor model.

We further assume that the alphabet $\mathcal{X}$ is the abelian group $\mathbb{Z}_q := \{0, \ldots, q-1\}$ and that

$$g_{ij}(x, x') = g(x, x') := \begin{cases} 1, & x = x', \\ -1, & x \neq x'. \end{cases} \tag{7.3}$$

Equations (7.1) to (7.3) define a (two-dimensional nearest-neighbor) Potts model.[1] (Some authors use the term *standard* Potts model to make explicit the distinction from the "clock" model.) To facilitate later discussions, we use f_{B} to denote $e^{-\beta E(x)}$ in (7.1) and refer to it as the "unnormalized Boltzmann distribution".

7.2.2 NFG Representation and Duality

A normal factor graph (NFG) $\mathcal{G}$ is a graph $(\mathcal{V}, \mathcal{E})$ where each edge $e \in \mathcal{E}$ is associated a variable x_e, and each vertex $v \in \mathcal{V}$ is associated a local function $f_v(x_{E(v)})$, where $E(v)$ is the set of edges incident with v, and for any set $\mathcal{A}$, $x_{\mathcal{A}} := \{x_a : a \in \mathcal{A}\}$. Let $\mathcal{X}_{\mathcal{G}}$ be the support of of the function defined as the multiplication of all local functions, and let $f_{\mathcal{G}}$ be the restriction of such function to $\mathcal{X}_{\mathcal{G}}$. Further, we define $Z_{\mathcal{G}}$ as the sum of $f_{\mathcal{G}}$ over $\mathcal{X}_{\mathcal{G}}$, and write $p_{\mathcal{G}} := f_{\mathcal{G}}/Z_{\mathcal{G}}$. Note that if all the local functions are nonnegative, then $p_{\mathcal{G}}$ is a probability distribution over $\mathcal{X}_{\mathcal{G}}$. In this case, in alignment with the previous discussions, we refer to $p_{\mathcal{G}}$, $f_{\mathcal{G}}$, and $Z_{\mathcal{G}}$ as the distribution, unnormalized distribution, and partition function of the NFG, respectively. We note that the above definitions of NFG and related terms deviate slightly from those in [1]. This is to simplify our presentation and exclude the concepts irrelevant to this chapter.

It is natural to associate with the model defined in Section 7.2.1 an NFG as in Fig. 7.1 (wrapping around is not shown). In the figure, each function marked by "=" is an "equality indicator function", namely, a function that evaluates to 1 if all it arguments are equal and evaluates to 0 otherwise; each equality indicator function corresponds to a random variable in the model. The function h in the figure is defined by $h(x, x') := e^{\beta g(x,x')}$. It is not hard to see that the unnormalized distribution, distribution and partition func-

[1] We slightly deviate from the traditional definition of the Potts model where the function g is usually assumed to take the value 0 instead of -1. Without altering the nature of the problem, this choice of function g includes the Ising model as the special case of $q = 2$.

tions associated with this NFG are respectively f_B, p_B and Z of the model defined by equations (7.1), (7.2) and (7.3).

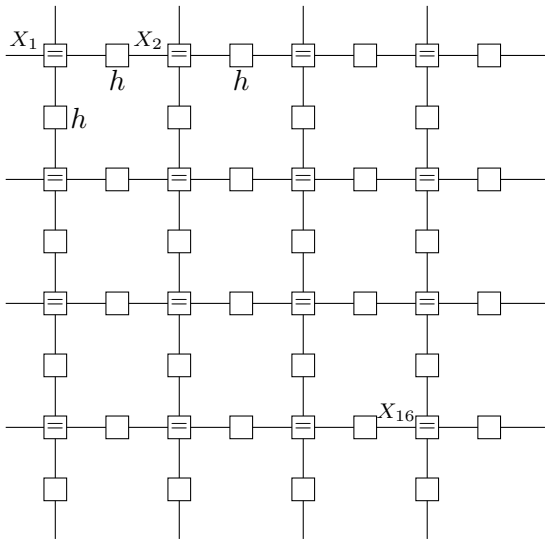


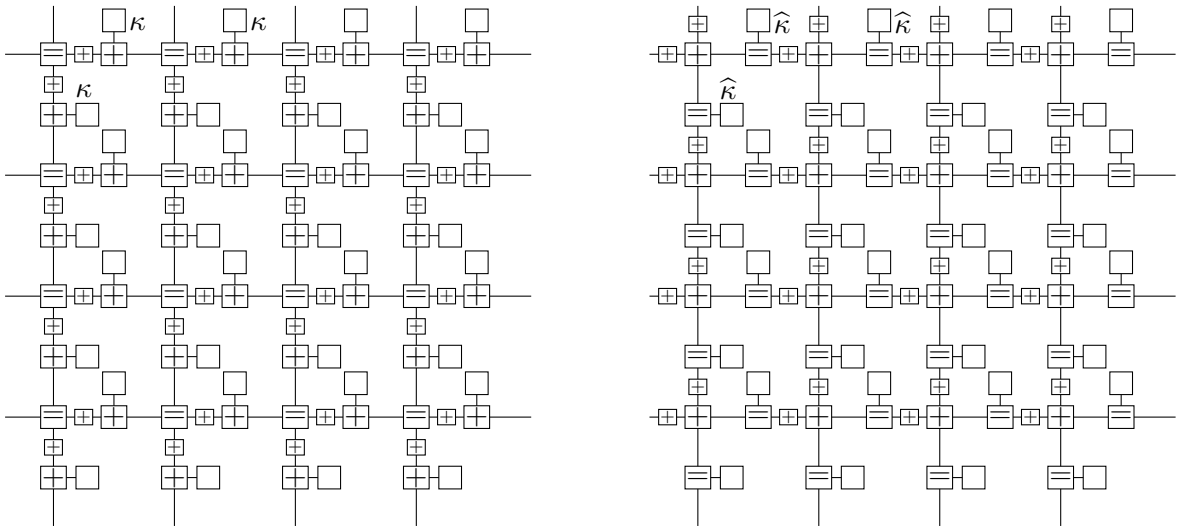
Figure 7.1: An NFG representing the model specified by (7.1), (7.2) and (7.3).

Noting that function g only depends on the difference between its arguments, we may express h by $h(x, x') := \kappa(x - x')$, where

$$\kappa(x) = \begin{cases} e^{\beta}, & x = 0, \\ e^{-\beta}, & x \neq 0. \end{cases} \tag{7.4}$$

Using function κ, the NFG in Fig. 7.1 may be converted to the NFG in Fig. 7.2 (a) without changing its unnormalized distribution, distribution and partition function. This latter NFG, which we denote by $\mathcal{G}$ is in fact preferred in the context of this work, since the results of this chapter depend crucially on a property of κ, which will become clear momentarily.

It is possible to introduce duality to NFG via the Fourier transform. Briefly, the Fourier transform of any function f on $\mathbb{Z}_q^m$ is another function $\widehat{f}$ defined on $\mathbb{Z}_q^m$. In particular, the Fourier transform of an equality indicator function is, up to scale, a "parity-check" indicator function, namely a function that evaluates to 1 if its argument sums to 0 and evaluates to 0 otherwise. A parity-check indicator function is marked by



(a) (b)

Figure 7.2: (a) The NFG $\mathcal{G}$ and (b) the dual NFG $\mathcal{G}'$.

"+" in an NFG. Further, the Fourier transform of the function κ is

$$\widehat{\kappa}(x) = \begin{cases} e^{\beta} + (q-1)e^{-\beta}, & x = 0 \\ e^{\beta} - e^{-\beta}, & x \neq 0. \end{cases} \tag{7.5}$$

Given an NFG, the dual NFG may be obtained by converting each local function to its Fourier transform and then inserting a parity-check indicator function to each edge. It can then be verified that the dual NFG of $\mathcal{G}$ is the NFG $\mathcal{G}'$ shown in Fig. 7.2 (b). A duality theorem (generalized Holant theorem) of NFG [1] states, in the context of our model, that $Z_{\mathcal{G}'} = Z_{\mathcal{G}}/q^N$.

7.2.3 Estimating Partition Function by Sampling NFG

Given an NFG $\mathcal{G}$ representing a statistical model, its partition function $Z_{\mathcal{G}}$ may be estimated via evaluating its unnormalized distribution $f_{\mathcal{G}}$ at a set of configurations $Y_1, Y_2, \ldots, Y_M$ randomly drawn from $\mathcal{X}_{\mathcal{G}}$. If these configurations are obtained by sampling the distribution $p_{\mathcal{G}}$ (which in practice can be done by Gibbs sampling), then the

Ogata-Tanemura (OT) [51] estimator can be defined as

$$Z_{\mathcal{G}}^{\mathrm{OT}}(M) := \frac{|\mathcal{X}_{\mathcal{G}}|}{\frac{1}{M}\sum_{i=1}^{M}\frac{1}{f_{\mathcal{G}}(Y_i)}}, \tag{7.6}$$

If these samples are drawn uniformly from $\mathcal{X}_{\mathcal{G}}$, an estimator, which we call the "uniform estimator", can be defined as

$$Z_{\mathcal{G}}^{\mathrm{U}}(M) := \frac{|\mathcal{X}_{\mathcal{G}}|}{M}\sum_{i=1}^{M} f_{\mathcal{G}}(Y_i), \tag{7.7}$$

It can be shown that as M increases, both $Z_{\mathcal{G}}^{\mathrm{OT}}(M)$ and $Z_{\mathcal{G}}^{\mathrm{U}}(M)$ converges to $Z_{\mathcal{G}}$.

Given the NFG $\mathcal{G}$ in Fig. 7.2 (a) that represents the Potts model, it is easy to see that every local function in the dual NFG $\mathcal{G}'$ in Fig. 7.2 (b) is non-negative. The dual NFG $\mathcal{G}'$ may then be regarded also as a statistical model and the above two estimators may be used to estimate the partition function $Z_{\mathcal{G}'}$, a scaled version of $Z_{\mathcal{G}}$. This technique was first used in [47] for Ising model (Potts model with $q = 2$), where the authors show empirically that at high temperature, both OT estimator and uniform estimator give more accurate estimates on the dual NFG.

7.3 Convergence Behaviour of the Estimators

Our analysis is primarily based on bounding the respective variances of the logarithm of the estimators for large M, as for any given number M of samples, such variance is an indicator of the estimation accuracy. Our development is largely in line with that of [57].

Given a statistical model NFG $\mathcal{G}$, it is possible to show

$$\begin{aligned}\lim_{M\to\infty} M\mathrm{Var}[\log(Z_{\mathcal{G}}^{\mathrm{OT}}(M))] &= \frac{Z_{\mathcal{G}}^2}{|\mathcal{X}_{\mathcal{G}}|^2}\mathrm{Var}\big[\frac{1}{f_{\mathcal{G}}(Y_1)}\big] \\ &= \frac{Z_{\mathcal{G}}}{|\mathcal{X}_{\mathcal{G}}|^2}\sum_{x\in\mathcal{X}_{\mathcal{G}}}\frac{1}{f_{\mathcal{G}}(x)} - 1. \end{aligned} \tag{7.8}$$

Proof: Let

$$X_M := \frac{\frac{1}{M}\sum_{i=1}^{M}\frac{1}{f_{\mathcal{G}}(Y_i)}}{|\mathcal{X}_{\mathcal{G}}|},$$

then

$$E[X_M] = \frac{1}{|\mathcal{X}_\mathcal{G}|} E[\frac{1}{f_\mathcal{G}(Y_1)}] = \frac{1}{|\mathcal{X}_\mathcal{G}|} \sum_x \frac{p_\mathcal{G}(x)}{f_\mathcal{G}(x)} = \frac{1}{Z_\mathcal{G}},$$

and

$$\mathrm{Var}[X_M] = \frac{1}{M|\mathcal{X}_\mathcal{G}|^2} \mathrm{Var}[\frac{1}{f_\mathcal{G}(Y_1)}].$$

From (7.6) we can rewrite $\log(Z_\mathcal{G}^{\mathrm{OT}})$ as

$$\log(Z_\mathcal{G}^{\mathrm{OT}}) = g(X_M),$$

where $g(x) := \log(\frac{1}{x}) = -\log(x)$, and so $g'(x) = \frac{-1}{x}$. Using Taylor expansion of g at $E[X_M]$,

$$\log(Z_\mathcal{G}^{\mathrm{OT}}) \simeq g(E(X_M)) + g'(E(X_M))(X_M - E(X_M)),$$

and so

$$\begin{aligned} \mathrm{Var}[\log(Z_\mathcal{G}^{\mathrm{OT}})] &\simeq \big(g'(E(X_M))\big)^2 \mathrm{Var}[X_M] \\ &= \frac{1}{(E[X_M])^2} \mathrm{Var}[X_M] \\ &= \frac{Z_\mathcal{G}^2}{M|\mathcal{X}_\mathcal{G}|^2} \mathrm{Var}[\frac{1}{f_\mathcal{G}(Y_1)}] \end{aligned}$$

The approximation is only valid in the limit, as g may be approximated as a linear function only when the variance of X_M is small. This method of first order approximation is often referred to as the *delta method*. For a more rigorous discussion on the delta method, see e.g. [12, Theorem 5.5.24]. ■

Similarly, it can be shown that

$$\begin{aligned} \lim_{M\to\infty} M\mathrm{Var}[\log Z_\mathcal{G}^{\mathrm{U}}(M)] &= \frac{|\mathcal{X}_\mathcal{G}|^2}{Z_\mathcal{G}^2} \mathrm{Var}[f_\mathcal{G}(Y_1)] \\ &= \frac{|\mathcal{X}_\mathcal{G}|}{Z_\mathcal{G}^2} \sum_{x\in\mathcal{X}_\mathcal{G}} f_\mathcal{G}^2(x) - 1. \end{aligned} \tag{7.9}$$

From this, the following proposition can be proved.

Proposition 5 *When sampling the NFG $\mathcal{G}$ of the Potts model,*

$$L_{\mathrm{OT}}(\beta) \leq \lim_{M\to\infty} M\mathrm{Var}[\log(Z_{\mathcal{G}}^{\mathrm{OT}}(M))] \leq R_{\mathrm{OT}}(\beta),$$
$$L_{\mathrm{U}}(\beta) \leq \lim_{M\to\infty} M\mathrm{Var}[\log(Z_{\mathcal{G}}^{\mathrm{U}}(M))] \leq R_{\mathrm{U}}(\beta),$$

where

$$L_{\mathrm{OT}}(\beta) := \frac{e^{2N\beta}}{|\mathcal{X}_{\mathcal{G}}|^2} - 1, R_{\mathrm{OT}}(\beta) := e^{4N\beta} - 1,$$
$$L_{\mathrm{U}}(\beta) := \frac{|\mathcal{X}_{\mathcal{G}}|}{(q + (|\mathcal{X}_{\mathcal{G}}| - q)e^{-8\beta})^2} - 1, R_{\mathrm{U}}(\beta) := e^{8N\beta} - 1.$$

Proof: We have

$$e^{-2N\beta} \leq f_{\mathcal{G}}(x) \leq e^{2N\beta}, \tag{7.10}$$
$$e^{-2N\beta} \leq \frac{1}{f_{\mathcal{G}}(x)} \leq e^{2N\beta}, \tag{7.11}$$

and so,

$$e^{2N\beta} \leq \quad Z_{\mathcal{G}} \quad \leq |\mathcal{X}_{\mathcal{G}}|e^{2N\beta}, \tag{7.12}$$
$$1 \leq \sum_x \tfrac{1}{f_{\mathcal{G}}(x)} \leq |\mathcal{X}_{\mathcal{G}}|e^{2N\beta}. \tag{7.13}$$

(The lower bound in (7.13) is trivially true, and made so to accommodate the case where the grid is of odd size while keeping the derived bounds simple. If the grid is of even size, it can be replaced with $e^{2N\beta}$— Color the grid in black and white such that no similar colors are adjacent. The lower bound in (7.12) is when all spins are equal— In fact there are q such configurations and one may replace the lower bound with $qe^{2N\beta}$.) Hence from (7.8),

$$\begin{aligned}\lim_{M\to\infty} M\mathrm{Var}[\log(Z_{\mathcal{G}}^{\mathrm{OT}}(M))] &= \frac{Z_{\mathcal{G}}}{|\mathcal{X}_{\mathcal{G}}|^2}\sum_{x\in\mathcal{X}_{\mathcal{G}}}\frac{1}{f_{\mathcal{G}}(x)} - 1 \\ &\overset{(7.13)}{\geq} \frac{Z_{\mathcal{G}}}{|\mathcal{X}_{\mathcal{G}}|^2} - 1 \\ &\overset{(7.12)}{\geq} \frac{e^{2N\beta}}{|\mathcal{X}_{\mathcal{G}}|^2} - 1,\end{aligned}$$

and

$$\begin{aligned}\lim_{M\to\infty} M\mathrm{Var}[\log(Z_{\mathcal{G}}^{\mathrm{OT}}(M))] &\overset{(7.13)}{\leq} \frac{Z_{\mathcal{G}}}{|\mathcal{X}_{\mathcal{G}}|^2}|\mathcal{X}_{\mathcal{G}}|e^{2N\beta} - 1 \\ &\overset{(7.12)}{\leq} \frac{|\mathcal{X}_{\mathcal{G}}|e^{2N\beta}}{|\mathcal{X}_{\mathcal{G}}|}e^{2N\beta} - 1 \\ &= e^{4N\beta} - 1.\end{aligned}$$

From (7.10) and (7.11), we also have

$$|\mathcal{X}_{\mathcal{G}}|e^{-2N\beta} \leq \quad Z_{\mathcal{G}} \quad \leq \big(q + (|\mathcal{X}_{\mathcal{G}}| - q)e^{-8\beta}\big)e^{2N\beta}, \tag{7.14}$$

$$e^{4N\beta} \leq \quad \textstyle\sum_x f_{\mathcal{G}}^2(x) \quad \leq |\mathcal{X}_{\mathcal{G}}|e^{4N\beta}. \tag{7.15}$$

(In (7.14) we needed a tighter upper bound than in (7.12). Instead of trivially replacing the summand with its largest value, we kept its largest q values and replaced the summand's remaining values with its second largest value.) From (7.9)

$$\begin{aligned}\lim_{M\to\infty} M\mathrm{Var}[\log(Z_{\mathcal{G}}^{\mathrm{U}}(M))] &= \frac{|\mathcal{X}_{\mathcal{G}}|}{Z_{\mathcal{G}}^2}\sum_{x\in\mathcal{X}_{\mathcal{G}}} f_{\mathcal{G}}^2(x) - 1. \\ &\overset{(7.15)}{\geq} \frac{|\mathcal{X}_{\mathcal{G}}|}{Z_{\mathcal{G}}^2}e^{4N\beta} - 1, \\ &\overset{(7.14)}{\geq} \frac{|\mathcal{X}_{\mathcal{G}}|e^{4N\beta}}{\big(q + (|\mathcal{X}_{\mathcal{G}}| - q)e^{-8\beta}\big)^2 e^{4N\beta}} - 1,\end{aligned}$$

and

$$\begin{aligned}\lim_{M\to\infty} M\mathrm{Var}[\log(Z_{\mathcal{G}}^{\mathrm{U}}(M))] &\overset{(7.15)}{\leq} \frac{|\mathcal{X}_{\mathcal{G}}|}{Z_{\mathcal{G}}^2}|\mathcal{X}_{\mathcal{G}}|e^{4N\beta} - 1, \\ &\overset{(7.14)}{\leq} \frac{|\mathcal{X}_{\mathcal{G}}|^2 e^{4N\beta}}{|\mathcal{X}_{\mathcal{G}}|^2 e^{-4N\beta}} - 1.\end{aligned}$$

■

We remark that the bounds presented in the proposition above (and later in Proposition 6) can be loose for some values of β. However, they suffice to explain the behaviour of the estimators on the primal and dual NFGs.

When β is small, say, in the order of N^{-m} for $m > 1$, both upper bounds R_{OT} and R_{U} in the proposition approach zero with increasing N. In this regime both estimators

provide good estimates of the partition function, without requiring asymptotically large M.

For large β, however, both estimators are inefficient. In particular, when $\beta > \log q$, the lower bound L_{OT} grows exponentially in N, which requires M to be at least exponential in N in order for the variance to be bounded within a constant. Similarly, when $\beta > \frac{\log q}{8} N$, the lower bound L_{U} also grows exponentially in N, making the uniform estimator inefficient. This is a rather exaggerated value of β, and we refer the reader to [43] for a better discussion on why the uniform estimator is inefficient for large β.

To get a better idea on relative performance between the OT and uniform estimators for large β, note that

$$\lim_{M\to\infty} M\mathrm{Var}[\log(Z^{\mathrm{U}}_{\mathcal{G}}(M))] \le |\mathcal{X}_{\mathcal{G}}| - 1,$$

which follows immediately from the fact that $\sum_x f^2_{\mathcal{G}}(x) \le Z^2_{\mathcal{G}}$. Comparing this upper bound with the lower bound L_{OT}, there exists $\beta_0 := \frac{3}{2}\log(q)$ above which the uniform estimator is more efficient than the OT estimator. This is in fact Theorem 2 of [57] for the model in this work.

On the dual side, we have the following bounds.

Proposition 6 *For any integer k, let $A_{k,\beta} := 1 + (k-1)e^{-2\beta}$, and let $r(\beta) := \frac{A_{q,\beta}}{A_{0,\beta}}$. When sampling the dual NFG $\mathcal{G}'$ for the Potts model (with N being an even number),*

$$L'_{\mathrm{OT}}(\beta) \le \lim_{M\to\infty} M\mathrm{Var}[\log(Z^{\mathrm{OT}}_{\mathcal{G}'}(M))] \le R'_{\mathrm{OT}}(\beta),$$
$$L'_{\mathrm{U}}(\beta) \le \lim_{M\to\infty} M\mathrm{Var}[\log(Z^{\mathrm{U}}_{\mathcal{G}'}(M))] \le R'_{\mathrm{U}}(\beta),$$

where

$$L'_{\mathrm{OT}}(\beta) := \frac{r^{2N}(\beta)}{|\mathcal{X}_{\mathcal{G}'}|^2} - 1, R'_{\mathrm{OT}}(\beta) := r^{2N}(\beta) - 1,$$
$$L'_{\mathrm{U}}(\beta) := \frac{|\mathcal{X}_{\mathcal{G}'}|}{\left(q + (|\mathcal{X}_{\mathcal{G}'}| - q)A_{0,\beta}\right)^2} - 1,$$
$$R'_{\mathrm{U}}(\beta) := r^{4N}(\beta) - 1.$$

Proof: We have

$$A_{0,\beta}^{2N} e^{2N\beta} \leq \quad f_{\mathcal{G}'}(x) \quad \leq A_{q,\beta}^{2N} e^{2N\beta}, \tag{7.16}$$

$$A_{q,\beta}^{-2N} e^{-2N\beta} \leq \quad \tfrac{1}{f_{\mathcal{G}'}(x)} \quad \leq A_{0,\beta}^{-2N} e^{-2N\beta}. \tag{7.17}$$

and so,

$$A_{q,\beta}^{2N} e^{2N\beta} \leq \qquad Z_{\mathcal{G}'} \qquad \leq |\mathcal{X}_{\mathcal{G}'}| A_{q,\beta}^{2N} e^{2N\beta}, \tag{7.18}$$

$$A_{0,\beta}^{-2N} e^{-2N\beta} \leq \quad \textstyle\sum_x \frac{1}{f_{\mathcal{G}'}(x)} \quad \leq |\mathcal{X}_{\mathcal{G}'}| A_{0,\beta}^{-2N} e^{-2N\beta}. \tag{7.19}$$

(The lower bound in (7.19) is valid since the model is of even size.) Hence from (7.8),

$$\begin{aligned}
\lim_{M\to\infty} M\mathrm{Var}[\log(Z_{\mathcal{G}'}^{\mathrm{OT}}(M))] &= \frac{Z_{\mathcal{G}'}}{|\mathcal{X}_{\mathcal{G}'}|^2} \sum_{x\in\mathcal{X}_{\mathcal{G}'}} \frac{1}{f_{\mathcal{G}'}(x)} - 1 \\
&\overset{(7.19)}{\geq} \frac{Z_{\mathcal{G}'}}{|\mathcal{X}_{\mathcal{G}'}|^2} A_{0,\beta}^{-2N} e^{-2N\beta} - 1 \\
&\overset{(7.18)}{\geq} \frac{A_{0,\beta}^{-2N} A_{q,\beta}^{2N}}{|\mathcal{X}_{\mathcal{G}'}|^2} - 1,
\end{aligned}$$

and

$$\begin{aligned}
\lim_{M\to\infty} M\mathrm{Var}[\log(Z_{\mathcal{G}'}^{\mathrm{OT}}(M))] &\overset{(7.19)}{\leq} \frac{Z_{\mathcal{G}'}}{|\mathcal{X}_{\mathcal{G}'}|^2} |\mathcal{X}_{\mathcal{G}'}| A_{0,\beta}^{-2N} e^{-2N\beta} - 1 \\
&\overset{(7.18)}{\leq} A_{q,\beta}^{2N} e^{2N\beta} A_{0,\beta}^{-2N} e^{-2N\beta} - 1 \\
&= A_{0,\beta}^{-2N} A_{q,\beta}^{2N} - 1.
\end{aligned}$$

From (7.16) and (7.17), we also have

$$|\mathcal{X}_{\mathcal{G}'}| A_{0,\beta}^{2N} e^{2N\beta} \leq Z_{\mathcal{G}'} \leq \big(q + (|\mathcal{X}_{\mathcal{G}'}| - q) A_{0,\beta}\big) A_{q,\beta}^{2N} e^{2N\beta}, \tag{7.20}$$

$$A_{q,\beta}^{4N} e^{4N\beta} \leq \sum_x f_{\mathcal{G}'}^2(x) \leq |\mathcal{X}_{\mathcal{G}'}| A_{q,\beta}^{4N} e^{4N\beta}, \tag{7.21}$$

where the upper bound in (7.20) follows from

$$Z_{\mathcal{G}'} \leq q A_{q,\beta}^{2N} e^{2N\beta} + (|\mathcal{X}_{\mathcal{G}'}| - q) A_{0,\beta} A_{q,\beta}^{2N-1} e^{(2N-1)\beta}.$$

From (7.9)

$$\begin{aligned}\lim_{M\to\infty} M\mathrm{Var}[\log(Z^{\mathrm{U}}_{\mathcal{G}'}(M))] &= \frac{|\mathcal{X}_{\mathcal{G}'}|}{Z^2_{\mathcal{G}'}}\sum_{x\in\mathcal{X}_{\mathcal{G}'}} f^2_{\mathcal{G}'}(x) - 1\\ &\overset{(7.21)}{\geq} \frac{|\mathcal{X}_{\mathcal{G}'}|}{Z^2_{\mathcal{G}'}} A^{4N}_{q,\beta} e^{4N\beta} - 1\\ &\overset{(7.20)}{\geq} \frac{|\mathcal{X}_{\mathcal{G}'}| A^{4N}_{q,\beta} e^{4N\beta}}{\big(q + (|\mathcal{X}_{\mathcal{G}'}| - q)A_{0,\beta}\big)^2 A^{4N}_{q,\beta} e^{4N\beta}} - 1\\ &= \frac{|\mathcal{X}_{\mathcal{G}'}|}{\big(q + (|\mathcal{X}_{\mathcal{G}'}| - q)A_{0,\beta}\big)^2} - 1.\end{aligned}$$

and

$$\begin{aligned}\lim_{M\to\infty} M\mathrm{Var}[\log(Z^{\mathrm{U}}_{\mathcal{G}'}(M))] &\overset{(7.21)}{\leq} \frac{|\mathcal{X}_{\mathcal{G}'}|}{Z^2_{\mathcal{G}'}} |\mathcal{X}_{\mathcal{G}'}| A^{4N}_{q,\beta} e^{4N\beta} - 1,\\ &\overset{(7.20)}{\leq} \frac{|\mathcal{X}_{\mathcal{G}'}|^2 A^{4N}_{q,\beta} e^{4N\beta}}{|\mathcal{X}_{\mathcal{G}'}|^2 A^{4N}_{0,\beta} e^{4N\beta}} - 1,\\ &= A^{-4N}_{0,\beta} A^{4N}_{q,\beta} - 1.\end{aligned}$$

■

When β is large, namely in the order of $\log(N)$, both upper bounds R'_{OT} and R'_{U} in the proposition approach zero with increasing N. In this regime both estimators provide good estimate of the partition function, without requiring asymptotically large M.

For small β, however, both estimators are inefficient. In particular, for $\beta < \frac{1}{2}\log\left(\frac{2q-1}{q-1}\right)$, the lower bound L'_{OT} grows exponentially in N, which requires M to be at least exponential in N in order for the variance to be bounded within a constant. Similarly, since $A_{0,\beta}$ approaches zero when β approaches zeros, L'_{U} becomes exponential in N.

Similar to the remark following Proposition 5, comparing $|\mathcal{X}_{\mathcal{G}'}| + 1$ with the lower bound L'_{OT}, it follows that there exists $\beta'_0 := \frac{1}{2}\log(1 + \frac{q}{q^2-1})$ *below* which the uniform estimator is more efficient than the OT estimator.

At this end, we have shown that on the dual NFG, the two estimators behave in an opposite trend (in β) to that on the primal NFG. It appears that such a phenomenon may fundamentally be related to a "duality" between "nearly uniform" and "nearly concentrated" distribution. More precisely, when both an NFG and its dual involve only

non-negative local functions, they both can be associated with a Boltzmann distribution. If one of the distributions is "nearly uniform", the other one is necessarily "nearly concentrated", namely, assigning most of the probability mass to only a few configurations. It is well-known in physics literature that the "near uniformity" and "near concentratedness" correspond respectively to high-temperature and low-temperature systems respectively. It appears that these sampling based estimators usually work well for high-temperature systems and work poorly for low-temperature systems. Taking an NFG to its dual, essentially reverts the "temperature".

7.4 Experiments

In this section we provide experimental results for the Potts model with $q = 4$ and grid-size $N = 10 \times 10$. We use the Gibbs sampling algorithm [50] on the primal and dual NFG to obtain samples from $p_{\mathcal{G}}$ and $p_{\mathcal{G}'}$, respectively. We estimate the log partition function per site (i.e. $\frac{1}{N}\log(Z)$), where depending on whether the primal or the dual NFG is used, the estimate of the partition function, which depends on the number of samples M, is defined as $\hat{Z}(M) := Z^{\mathrm{OT}}_{\mathcal{G}}(M)$ and $\hat{Z}(M) := q^N Z^{\mathrm{OT}}_{\mathcal{G}'}(M)$, respectively. (Similar definitions are used for the uniform estimator.) For any number of samples M, we repeat the experiment 30 times and record the value of $\frac{1}{N}\log(\hat{Z}_i(M)), i = 1, \ldots, 30$, where for each trial i, the initial configuration is chosen independently and according to the uniform distribution. The "quality" of the estimation at any M is decided based on the standard deviation of the trials from their mean (with respect to the uniform distribution on the set $\{1, \ldots, 30\}$).

Figs 7.3 and 7.4 show the estimated log partition function per site, i.e., $\log(\hat{Z}(M))/N$, for the low temperature $\beta = 1.2$ Fig. 7.3 shows the estimation based on the primal NFG using both the uniform estimator (left) and the OT estimator (right). Using up to 10^6 samples, both estimators fail to converge, and so do not provide a good estimation. This can also be seen in the dashed lines in Fig. 7.5 showing the standard deviation of the uniform estimator (left) and the OT estimator (right), where the standard deviation in both cases remains high. In contrast, Fig. 7.4 shows fast convergence of the estimators

on the dual NFG. The standard deviation of the estimations obtained from the dual NFG is shown using the solid lines in Fig. 7.5. Fig. 7.6 shows the standard deviation of the estimations for the high temperature of $\beta = 0.18$. In this case estimations based on the primal NFG have a lower standard deviation compared to the dual NFG, and so provide a better estimation. In Fig. 7.7 (a), showing the standard deviation versus β using uniform sampling, one observes the behaviour of the estimator versus β as discussed in Section 7.3.

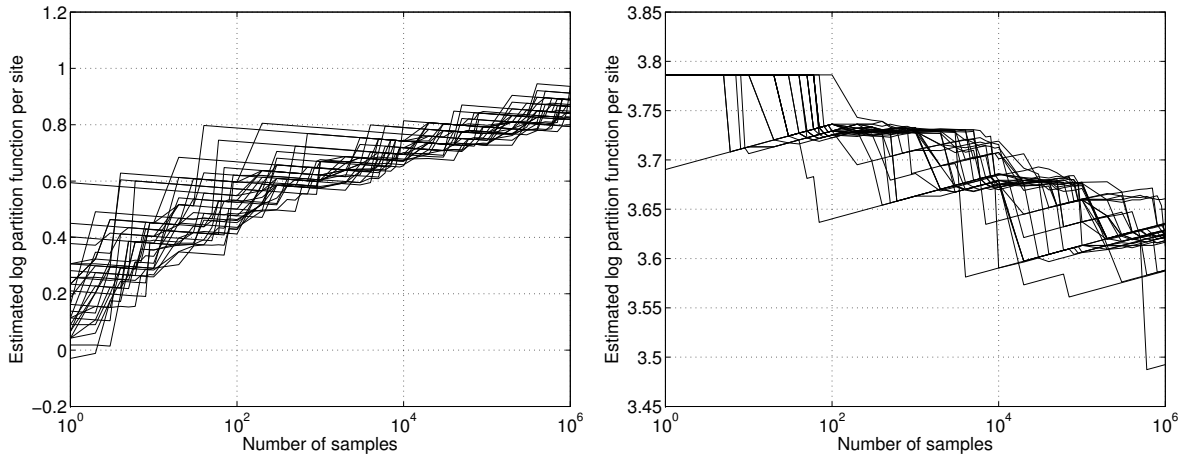


Figure 7.3: Potts model at low temperature $\beta = 1.2$ using the primal NFG. The two figures show the estimated log partition function per site versus the number of samples using the uniform estimator (left) and the OT estimator (right), where each line represents a trial.

7.5 Concluding Remarks: Beyond the Potts model

This chapter shows analytically and experimentally that stochastic estimators of partition functions exhibit opposite trends on NFG representation of a model and its dual. As remarked in Section 7.3, this phenomenon is fundamentally related to a duality between "nearly concentrated" and "nearly uniform" distribution. This understanding allows the results presented above to extend beyond the Potts models. In particular, one may consider two-dimensional nearest neighbor models whose bivariate local function is of the

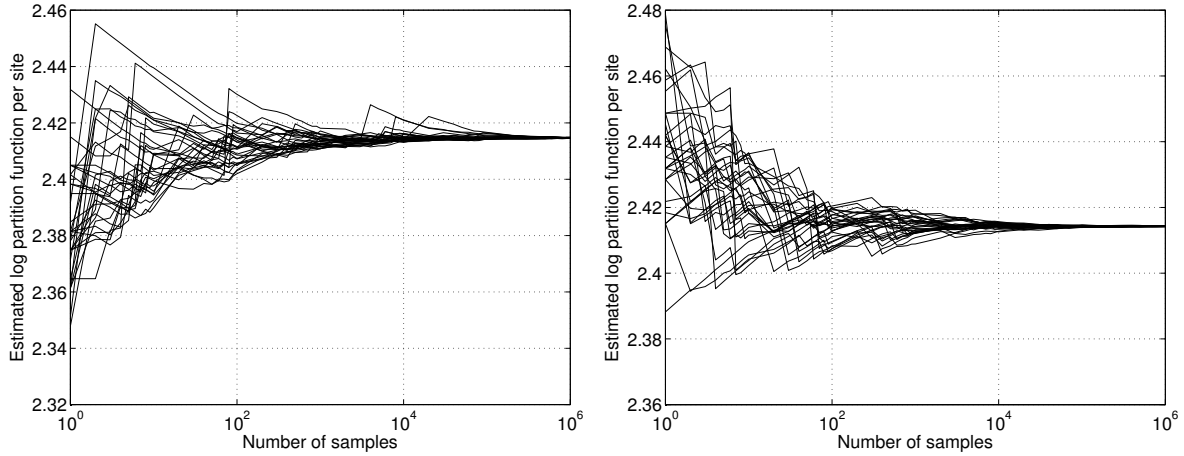


Figure 7.4: Potts model at low temperature $\beta = 1.2$ using the dual NFG. The two figures show the estimated log partition function per site versus the number of samples using the uniform estimator (left) and the OT estimator (right), where each line represents a trial.

form $h(x, x') := \kappa(x - x')$ for other functions κ. When both κ and $\widehat{\kappa}$ are a non-negative real function, the duality between uniformity and concentratedness is expected to hold and such a phenomenon is expected to occur. As an example, consider the "clock model," which is defined in the same way as the Potts model under the choice

$$\kappa_{\text{clock}}(x) = e^{\beta \cos(2\pi x/q)}, \tag{7.22}$$

for all $x \in \mathbb{Z}_q$. (Hence, it is within the scope of models of Fig. 7.2 (a).) From Lemma 14 below, $\widehat{\kappa}_{\text{clock}}$ is a positive function, and so it is possible to take the dual NFG route toward estimating its partition function.

Lemma 14 $\widehat{\kappa}_{\text{clock}}$ *is a positive function.*

Proof: For any $x, y \in \mathbb{Z}_q$, let $\chi_y(x) := e^{2\pi\sqrt{-1}xy/q}$. Using Taylor expansion, we have

$$\kappa_{\text{c}}(x) = \sum_{n=0}^{\infty} \frac{\beta^n g_n(x)}{n!},$$

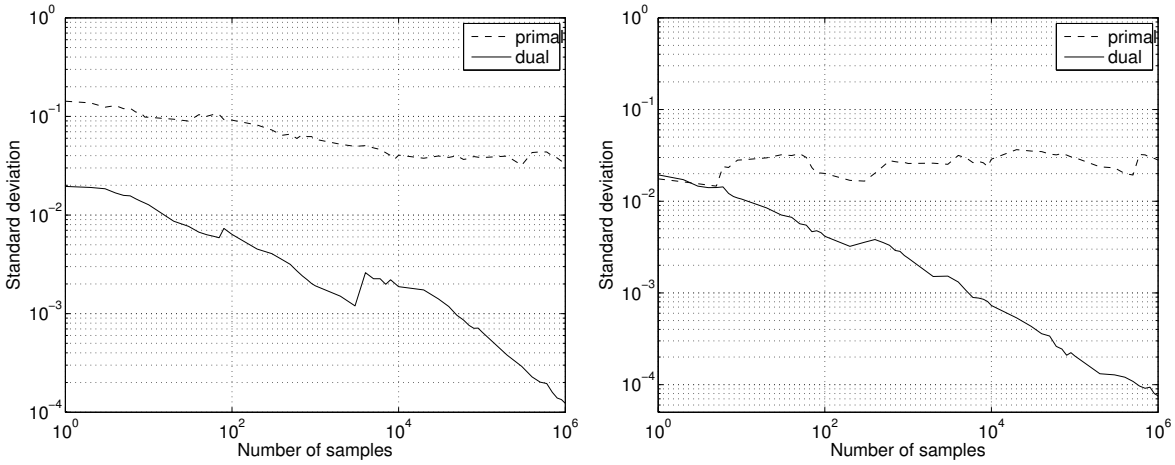


Figure 7.5: Potts model at low temperature $\beta = 1.2$, where the standard deviation of the estimated log partition function per site is shown for the uniform (left) and OT (right) estimators based on the primal (dashed line) and dual (solid line) NFGs.

where

$$\begin{aligned} g_n(x) := \cos^n(2\pi x/q) &= \frac{1}{2^n}\big(\chi_1(x) + \chi_1(-x)\big)^n \\ &= \frac{1}{2^n}\sum_{l=0}^{n}\binom{n}{l}\chi_1^{n-l}(x)\chi_1^{l}(-x) \\ &= \frac{1}{2^n}\sum_{l=0}^{n}\binom{n}{l}\chi_{\overline{n}-2\overline{l}}(x), \end{aligned}$$

where for any $m \in \mathbb{Z}$, $\overline{m} \in \mathbb{Z}_q$ is defined as m modulo q. Hence,

$$\widehat{g_n}(\chi_k) = \frac{q}{2^n}\sum_{l=0}^{n}\binom{n}{l}[\overline{n} + k - 2\overline{l} = 0]$$

is a non-negative function that is upper bounded by q, where for any m, $[m = 0]$ is the indicator function evaluating to one iff $m = 0$. Therefore,

$$\widehat{\kappa}_{\mathrm{c}}(\chi_k) = \sum_{n=0}^{\infty}\frac{\beta^n \widehat{g_n}(\chi_k)}{n!}$$

is a positive function for $\beta > 0$. (This follows since for any $k \in \mathbb{Z}_q$, $\widehat{g_n}(\chi_k)$ cannot be zero for all n. In particular, $\widehat{g_{q-k}}(\chi_k) > 0$.) Finally, the series in the RHS is convergent since

$$\sum_{n=0}^{\infty}\frac{\beta^n \widehat{g_n}(\chi_k)}{n!} \le q\sum_{n=0}^{\infty}\frac{\beta^n}{n!} = qe^{\beta}.$$

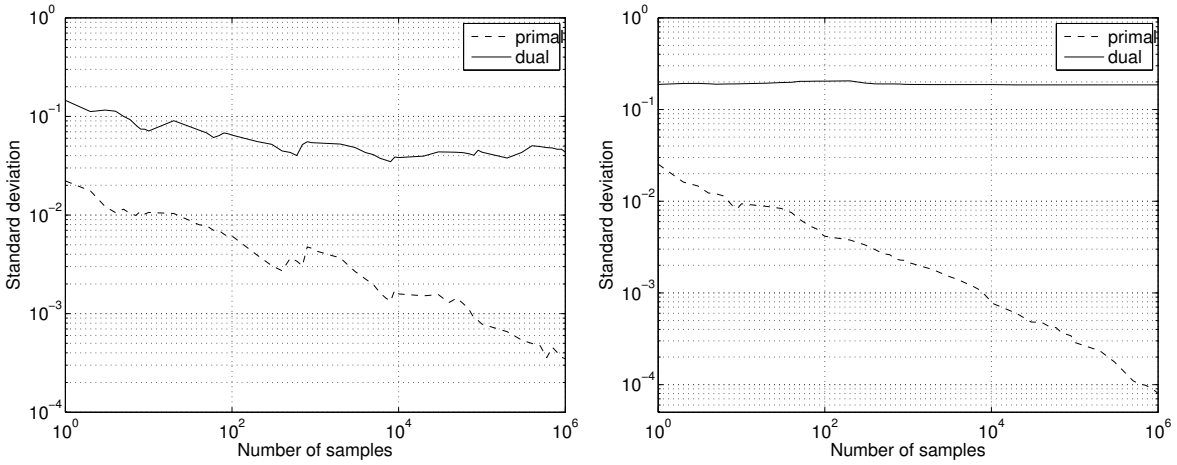


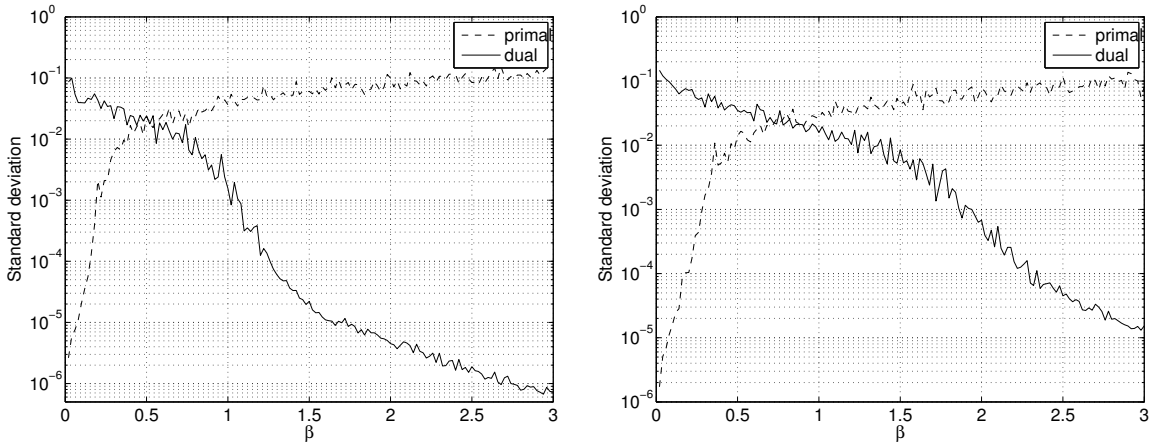
Figure 7.6: Potts model at high temperature $\beta = 0.18$, where the standard deviation of the estimated log partition function per site is shown for the uniform (left) and OT (right) estimators based on the primal (dashed line) and dual (soled line) NFGs.

■

Consider for instance the clock model with $q = 4$. It is not hard to see that $p_{\mathcal{G}}$ in this case is a "concentrated" distribution for low temperatures and an "almost uniform" distribution for high temperatures. From this and the fact that

$$\widehat{\kappa}_{\text{clock}}(x) = \begin{cases} e^{\beta} + e^{-\beta} + 2, & x = 0, \\ e^{\beta} - e^{-\beta}, & x \in \{1, 3\}, \\ e^{\beta} + e^{-\beta} - 2, & x = 2, \end{cases} \tag{7.23}$$

one may obtain similar results to Propositions 5 and 6. Simulation results for this model are shown in Fig. 7.7 (b).



(a) Potts model.

(b) Clock model.

Figure 7.7: Standard deviation of the estimated free energy per site versus β using uniform sampling with $M = 10^6$.

Chapter 8

Conclusions

Sums of products are fundamental in physics, computer science, coding theory, information theory, and indeed many other fields. In this thesis, we introduced the exterior function semantics, which allows a normal factor graph representation of "normal" sum-of-products forms. The normality constraint requires that the sum-of-products form is such that any variable must appear in at most two functions. This is a nonrestrictive constraint since any sum-of-products form can be normalized without introducing any additional complexity. Within the framework of normal factor graphs and the exterior function semantics, we introduced the notion of holographic transformation, and we provided a theorem, the generalized Holant theorem, which relates a pair of transformed normal factor graphs. As corollaries of this theorem, one may recover Valiant's original Holant theorem on one hand, and on the other, obtain a normal factor graph duality result, a special case of which is Forney's duality theorem of normal graphs. These are the basis of this thesis and were laid down in Chapters 2 and 3.

Several notions from linear algebra may be viewed as sum-of-products forms, which makes normal factor graphs, under the exterior function semantics, a viable graphical tool for proving some results in linear algebra, similar to that of trace diagrams. We addressed this in Chapter 4 and showed that normal factor graphs may be used to prove some elementary results from linear algebra. The discussion was focused on how normal factor graphs can be used for such purpose, and it remains open whether normal factor

graphs may facilitate the proofs of some harder results in linear algebra. A similar approach in Chapter 6 was adopted towards the discussion of translation association schemes.

Normal factor graphs as probabilistic models were studied in Chapter 5. In particular, two models were presented, the constrained and the generative model. While a constrained model is equivalent to a factor graph, many existing probabilistic models; namely, convolutional factor graphs, linear characteristic model, and cumulative distribution networks; may be obtained from the generative model via a specification of the interface functions and, if needed, a proper holographic transformation. The relation, if any, between normal factor graphs and other graphical models such as chain graphs and ancestral graphs is open, and might be an interesting research problem.

As an application of the normal factor graph duality beyond Forney's duality, in Chapter 7 we looked at the estimation of the partition function of the two-dimensional nearest-neighbor Potts model. Using analytical and experimental results, we showed that the Ogata-Tanemura and uniform estimators are more efficient on the dual normal factor graph at low temperature, and are more efficient on the primal normal factor graph at high temperature. We point out sufficient conditions for such trend to hold, however, a more refined study of the topic to determine necessary and sufficient conditions, and hence extend the results beyond Potts model, is an interesting direction of investigation.

Appendix A

Converting Arbitrary Sum-of-Products Forms to NFGs

In the framework of factor graphs [38], the product of any collection of multivariate functions may be represented by a factor graph. By specifying a subset of the variables in the factor graph to be "internal" (namely, to be summed over), it is then possible to represent *any* sum-of-products form using a factor graph with additional marks on some variable vertices.

Figure A.1 is an example of a sum-of-products form represented by such a "marked" factor graph, where the variable vertices marked with "$\times$" represent internal variables; the variable vertices without such marks are external variables, namely, those remaining in the argument of the represented function. Such a "marked" factor graph then represents the product of all local functions with all internal variables summed over; the function resulting from the summation then clearly involves only the external variables, analogous to the exterior function of an NFG.

Since a factor graph can have an unrestricted topology and an arbitrary subset of its variable vertices may be marked, it is possible to represent any sum-of-products form using a "marked" factor graph.

Without loss of generality, we assume that there are no degree-1 internal variable vertices in a "marked" factor graph, since otherwise it is always possible to modify the

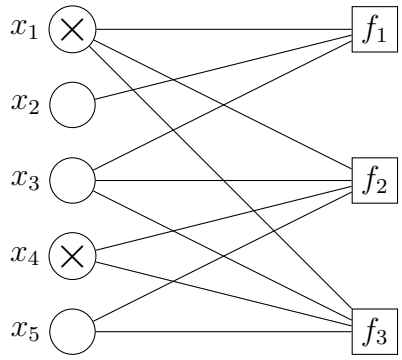
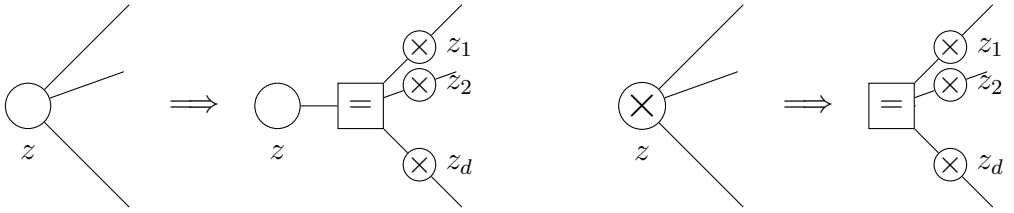


Figure A.1: The "marked" factor graph representing the sum-of-products form $\sum_{x_1,x_4} f_1(x_1,x_2,x_3)f_2(x_1,x_3,x_4,x_5)f_3(x_1,x_3,x_4,x_5)$.



(a) Replicating an external variable

(b) Replicating and internal variable

Figure A.2: Variable Replication Procedure.

local function connecting to the variable by summing over that variable.

The following procedure, operating on a "marked" factor graph representations, "normalizes" any sum-of-products form.

Variable Replication Procedure. Suppose that a variable z in a factor graph has degree d. Then we may create d replicas $\{z_1, z_2, \ldots, z_d\}$ of variable z, isolate z from its edges, and attach each of the d replicas to one of these edges. Remove z if z is an internal variable in the original factor graph. Connect all the replicas of z and z itself, if it is kept, to a new function vertex representing $\delta_=(\cdot)$. Finally, mark all replicas of z internal (i.e., with "$\times$"). Figure A.2 is a graphical illustration of this procedure.

A procedure similar to the Vertex Replication Procedure above was first presented

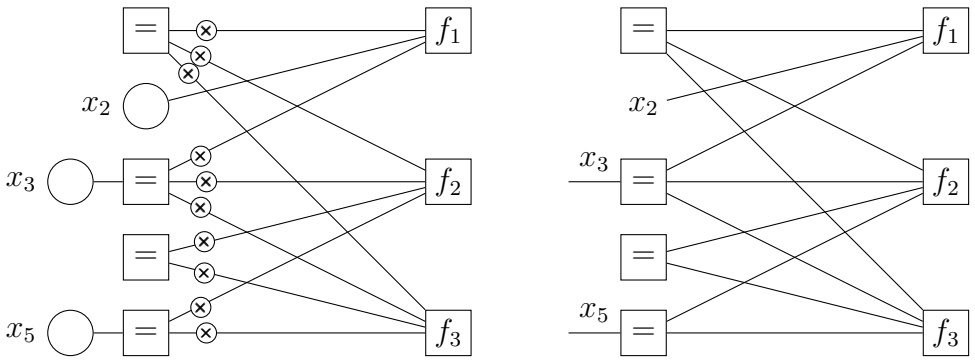


Figure A.3: The sum-of-products form resulting from normalizing the "marked" factor graph of Figure A.1. Left: the sum-of-products form represented as a "marked" factor graph; right: the sum-of-product form represented as an NFG.

in [23]. It is easy to verify that when applying the Variable Replication Procedure to any variable vertex, the sum-of-products form corresponding to the resulting "marked" factor graph expresses the same function as the original sum-of-products form does.

On a "marked" factor graph, we may apply this procedure to every variable that does not satisfy the "normal" degree restriction (namely that an internal variable vertex have degree two and an external variable vertex have degree one). It is straightforward to verify that in the resulting "marked" factor graph, the normal degree restriction is necessarily satisfied by all variables. We can then represent the resulting sum-of-products form using the NFG notation, representing internal variables as edges and external variables as dangling edges.

Figure A.3 shows the sum-of-products form resulting from normalizing the "marked" factor graph in Figure A.1.

Appendix B

Translation Schemes

Given a d-class translation scheme $(G, \mathfrak{L})$, we have $\mathcal{A} := \text{span}\{L_i : i \in \mathcal{D}\}$ and $\hat{A} := \{\hat{f} : f \in \mathcal{A}\}$.

B.1 Dual translation scheme

This proof is due to [11]. In this section, we show that $\hat{A}$ is the algebra of a translation scheme. Before we start, note that $\hat{\mathcal{A}}$ is a vector space that contains the constant function $\mathbf{1}$ since $L_0 \in \mathcal{A}$. First we show that $\hat{\mathcal{A}}$ exhibits a basis of indicator functions. Let $f \in \hat{\mathcal{A}}$ be a function with the largest number of distinct values, say it has $m+1$ distinct values, and denote such values by $\lambda_0, \ldots, \lambda_m$. (The labelling is chosen such that $f(0) = \lambda_0$, and without loss of generality, we may assume $\lambda_i \neq 0$ for all i since $f + c = f + c\mathbf{1} \in \hat{\mathcal{A}}$ for any $c \in \mathbb{C}$.) Hence, we can write f as $f = \lambda_0 L'_0 + \cdots + \lambda_m L'_m$, where $L'_i(x) := [f(x) = \lambda_i]$ for all $x \in \widehat{G}$ and all i. For all i, we have $f - \lambda_i = f - \lambda_i \mathbf{1} \in \hat{\mathcal{A}}$, and so for any j, it holds that $g_j := \prod_{i \neq j}(f - \lambda_i) \in \hat{\mathcal{A}}$. Noting that g_j is equal to L'_j, up to the scaling factor $\prod_{i \neq j}(\lambda_j - \lambda_i)$, it follows that $L'_j \in \hat{\mathcal{A}}$. Clearly, the functions L'_i are independent, and so $m = d$, and the set $\{L'_i : i \in \mathcal{D}\}$ forms a basis of $\hat{\mathcal{A}}$.

Now we verify Definition 1. Assume L'_0 is not the indicator function $[x = 0]$. Since $\sum_i L_i = \mathbf{1} \in \mathcal{A}$, the function $[x = 0]$ is in $\hat{\mathcal{A}}$. Hence, the function $f - \lambda_0[x = 0]$ is in $\hat{\mathcal{A}}$ and it has $m + 2$ distinct values, contradicting the maximality of f. Closure under

convolution is immediate since $\mathcal{A}$ is closed under multiplication. Hence, $(\widehat{G}, \mathfrak{L}')$ is a translation scheme, where $\mathfrak{L}'$ is the map from $\widehat{G}$ to $\mathcal{D}$ defined as $\mathfrak{L}'(x) = i$ if $x \in \mathcal{L}'_i$.

B.2 On the definition of a translation association scheme

Here we show that if $(G, \mathfrak{L})$ is a d-translation scheme as in Definition 1, then for any $i \in \mathcal{D}$, it must be the case that $L_i^- = L_j$ for some $j \in \mathcal{D}$.

We have $\mathcal{A} := \text{span}\{L_i : i \in \mathcal{D}\}$ and from Section B.1 $\hat{\mathcal{A}} = \text{span}\{L'_i : i \in \mathcal{D}\}$, where L'_i are real (indicator functions). First we show that $\mathcal{A}$ is closed under Hermitian conjugate. We have

$$\begin{aligned} f \in \mathcal{A} \;&\Rightarrow\; \hat{f} = \sum_i \lambda_i L'_i \in \hat{\mathcal{A}} \\ &\Rightarrow\; \overline{\hat{f}} = \sum_i \overline{\lambda}_i L'_i \in \hat{\mathcal{A}} \\ &\Rightarrow\; \overline{f^-} \in \mathcal{A}, \end{aligned}$$

where the second implication is because L'_i is real. Hence, if $f \in \mathcal{A}$ is real, we have $f^- \in \mathcal{A}$. Therefore, we have $\left\{L_i^- : i \in \mathcal{D}\right\} \subset \mathcal{A}$. In fact, this set is a basis of indicator functions as can be easily seen from:

$$\alpha_0 L_0^- + \cdots + \alpha_d L_d^- = 0 \Leftrightarrow \alpha_0 L_0 + \cdots + \alpha_d L_d = 0.$$

Since $\{L_i : i \in \mathcal{D}\}$ and $\left\{L_i^- : i \in \mathcal{D}\right\}$ are two bases of indicator functions, the change of bases matrix must have $\{0, 1\}$ entries. Since the Hermitian conjugate (viewed as a map $\mathcal{A} \to \mathcal{A}$) preserves the size of the support of each function, such change of bases matrix must be a permutation matrix.

Bibliography

[1] A. Al-Bashabsheh and Y. Mao. Normal factor graphs and holographic transformations. *IEEE Trans. Inf. Theory*, 57(2):752–763, Feb. 2011.

[2] A. Al-Bashabsheh, Y. Mao, and T. Chan. Translation schemes and LP bounds. In *13th Canadian Workshop on Inf. Theory*, pages 200–204, Toronto, Canada, June 2013.

[3] A. Al-Bashabsheh, Y. Mao, and P. O. Vontobel. Normal factor graphs: A diagrammatic approach to linear algebra. In *Proc. IEEE Int. Symp. Inf. Theory*, pages 2178–2182, Saint-Petersburg, Russia, July 2011.

[4] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. Inf. Theory*, 20(2):284–287, March 1974.

[5] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. In *IEEE Int. Conf. on Commun.*, pages 1064–1070, Geneva, Switzerland, May 1993.

[6] D. Bickson and C. Guestrin. Inference with multivariate heavy-tails in linear models. In *Neural Information Processing System (NIPS)*, pages 208–216, Vancouver, Canada, Dec. 2010.

[7] A. E. Brouwer, A. M. Cohen, and A. Neumaier. *Distance-Regular Graphs*. Springer-Verlag, 1989.

[8] J. Cai, P. Lu, and M. Xia. Holographic algorithms with matchgates capture precisely tractable planar #CSP. In *51st Annual IEEE Symp. on Foundations of Computer Science*, pages 427–436, Las Vegas, NV, USA, Oct. 2010.

[9] J. Y. Cai and P. Lu. Holographic algorithms: From art to science. In *Proc. of the 39th Annual ACM Symp. on Theory of Computing*, pages 401–410, San Diego, CA, USA, June 2007.

[10] J. Y. Cai, P. Lu, and M. Xia. Holographic algorithms by Fibonacci gates and holographic reductions for hardness. In *49th Annual IEEE Symp. on Foundations of Computer Science*, pages 644–653, Philadelphia, PA, USA, Oct. 2008.

[11] P. Camion. Codes and association schemes: Basic properties of association schemes relevant to coding. *Handbook of Coding Theory*, 2:1441–1567, 1998.

[12] G. Casella and R. L. Berger. *Statistics Inference*. Duxbury, Pacific Grove, California, 2002.

[13] T. H. Chan, A. Grant, and T. Britz. Quasi-uniform codes and their applications. *IEEE trans. Inf. Theory*, 59(12):7915–7926, Nov. 2013.

[14] T. H. Chan and Y. Mao. Entropic interpretation to Delsarte's LP bound. Personal communication, 2013.

[15] V. Y. Chernyak and M. Chertkov. Loop calculus and belief propagation for q-ary alphabet: Loop tower. In *Proc. IEEE Int. Symp. Inf. Theory*, pages 316–320, Nice, France, June 2007.

[16] V. Y. Chernyak and M. Chertkov. Planar graphical models which are easy. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(11), Nov. 2010.

[17] M. Chertkov and V. Y. Chernyak. Loop calculus in statistical physics and information science. *Physical Review E*, 73(6), June 2006.

[18] M. Chertkov and V. Y. Chernyak. Loop series for discrete statistical models on graphs. *Journal of Statistical Mechanics: Theory and Experiment*, 2006, June 2006.

[19] P. Cvitanović. *Group Theory: Birdtracks, Lie's, and Exceptional Groups*. Princeton Univ. Press, 2008.

[20] P. Delsarte. An algebraic approach to the association schemes of coding theory. *Philips Research Reports Supplement*, 1973.

[21] P. Delsarte and V. I. Levenshtein. Association schemes and coding theory. *IEEE Trans. Inf. Theory*, 44(6):2477–2504, Oct. 1998.

[22] G. D. Forney, Jr. Transforms and groups. In A. Vardy, editor, *Codes, Curves and Signals: Common Threads in Communications*, chapter 7. Norwood, MA: Kluwer, 1998.

[23] G. D. Forney, Jr. Codes on graphs: Normal realizations. *IEEE Trans. Inf. Theory*, 51(2):520–548, Feb. 2001.

[24] G. D. Forney, Jr. Principles of digital communication II. Printed notes for MIT course 6.451, 2005.

[25] G. D. Forney, Jr. Codes on graphs: Duality and MacWilliams identities. *IEEE Trans. Inf. Theory*, 57(3):1382–1397, March 2011.

[26] G. D. Forney, Jr. and P. O. Vontobel. Partition functions of normal factor graphs. In *Proc. Inf. Theory and Applications Workshop*, San Diego, CA, Feb. 2011.

[27] G. D. Forney Jr. The viterbi algorithm. In *Proc. IEEE*, volume 61, pages 268–278, March 1973.

[28] R. G. Gallager. Low-density parity-check codes. *IRE Trans. Info. Theory*, 8(1):21–28, Jan. 1962.

[29] C. Godsil. Generalized hamming schemes. 2010. Available at arXiv:1011.1044.

[30] J. C. Huang and B. J. Frey. Cumulative distribution networks and the derivative-sum-product algorithm. In *Proc. of the 24th conference on Uncertainty in Artificial Intelligence*, Helsinki, Finland, July 2008.

[31] J. C. Huang and B. J. Frey. Cumulative distribution networks and the derivative-sum-product algorithm: Models and inference for cumulative distribution functions on graphs. *The Journal of Machine Learning Research*, 12:301–348, Jan. 2011.

[32] M. I. Jordan. *An introduction to probabilistic graphical models*. 2002. Unpublished draft.

[33] P. W. Kasteleyn. The statistics of dimers on a lattice. *Physica*, 27(12):1209–1225, Dec. 1961.

[34] R. Koetter. On the representation of codes in Forney graphs. In R. E. Blahut and R. Koetter, editors, *Codes, Graphs, and Systems*, pages 425–450. Kluwer Academic Publishers, Feb. 2002.

[35] R. Koetter, M. Effros, T. Ho, and M. Médard. Network codes as codes on graphs. In *Proc. 38th Annual Conf. on Information Sciences and Systems*, pages 1–6, Princeton, NJ, USA, March 2004.

[36] R. Koetter and A. Vardy. Factor graphs: Constructions, classification, and bounds. In *Proc. IEEE Int. Symp. Inf. Theory*, Cambridge, MA, USA, Aug. 1998.

[37] F. R. Kschischang and B. J. Frey. Iterative decoding of compound codes by probability propagation in graphical models. *IEEE J. Select. Areas Commun.*, 16:219–230, Feb. 1998.

[38] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory*, 47(2):498–519, Feb. 2001.

[39] S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[40] H.-A. Loeliger. An introduction to factor graphs. *IEEE Sig. Proc. Mag.*, 21(1):28–41, Jan. 2004.

[41] H.-A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F. R. Kschischang. The factor graph approach to model-based signal processing. *Proc. IEEE*, 95(6):1295–1322, June 2007.

[42] D. J. C. MacKay and R. M. Neal. Good codes based on very sparse matrices. In *Cryptography and coding*, volume 1025 of *Lecture notes in computer science*, pages 100–111, 1995.

[43] D. J. MacKay. Introduction to Monte Carlo methods. In *Learning in graphical models*, volume 89, pages 175–204, 1998.

[44] Y. Mao and F. R. Kschischang. On factor graphs and the Fourier transform. *IEEE Trans. Inf. Theory*, 51(5):1635–1649, 2005.

[45] Y. Mao, F. R. Kschischang, and B. J. Frey. Convolutional factor graphs as probabilistic models. In *Proc. of the 20th conference on Uncertainty in artificial intelligence*, pages 374–381, Banff, Canada, 2004.

[46] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng. Turbo decoding as an instance of Pearl's belief propagation algorithm. *IEEE J. Select. Areas Commun.*, 16(2):140–152, Feb. 1998.

[47] M. Molkaraie and H.-A. Loeliger. Partition function of the Ising model via factor graph duality. In *Int. Symp. Inf. Theory*, pages 2304–2308, Istanbul, Turkey, July 2013.

[48] S. Morse and E. Peterson. Trace diagrams, signed graph coloring and matrix minors. *Involve, a Journal of Mathematics*, 3(1):33–66, 2010.

[49] R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Dept. Comp. Science, University of Toronto, 1993.

[50] M. Newman and G. Barkema. *Monte Carlo methods in statistical physics*. Oxford University Press, Oxford, Reprint 2001.

[51] Y. Ogata and M. Tanemura. Estimation of interaction potentials of spatial point patterns through the maximum likelihood procedure. *Ann. Inst. Statist. Math.*, 33(1):315–338, 1980.

[52] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[53] R. Penrose. *Combinatorial Mathematics and Its Applications, chapter Applications of negative dimensional tensors.* Academic Press, 1971.

[54] R. Penrose. *The Road to Reality: A Complete Guide to the Laws of the Universe.* Jonathan Cape, London, 2004.

[55] E. Peterson. On a diagrammatic proof of the Cayley-Hamilton theorem. 2009. Available at arXiv:0907.2364v1.

[56] E. Peterson. Unshackling linear algebra from linear notation. 2009. Available at arXiv: 0910.1362.

[57] G. Potamianos and J. Goutsias. Stochastic approximation algorithms for partition function estimation of Gibbs random fields. *IEEE Trans. Inf. Theory*, 43(6):1948–1965, Nov. 1997.

[58] M. Schwartz and J. Bruck. Constrained codes as networks of relations. *IEEE Trans. Inf. Theory*, 54(5):2179–2195, May 2008.

[59] R. Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inf. Theory*, 27(5):533–547, Sept. 1981.

[60] H. N. V. Temperley and M. E. Fisher. Dimer problem in statistical mechanics— An exact result. *Philosophical Magazine*, 6(68):1061–1063, 1961.

[61] L. G. Valiant. Holographic algorithms (extended abstract). In *Proc. 45th Annual IEEE Symp. on Foundations of Computer Science*, pages 306–315, Rome, Italy, Oct. 2004.

[62] L. G. Valiant. Some observations on holographic algorithm. In *Proc. 9th Latin American Theoretical Informatics*, pages 577–590, Oaxaca, Mexico, April 2010. Springer Berlin Heidelberg.

[63] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory*, 13(2):260–269, April 1967.

[64] P. O. Vontobel. Kalman filters, factor graphs, and electrical networks. *Post-Diploma Project, ETH Zurich*, 2002.

[65] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. *IEEE Trans. Inf. Theory*, 51(7):2313–2335, July 2005.

[66] N. Wiberg. *Codes and decoding on general graphs*. PhD thesis, Univ. Linköping, Linköping, Sweden, 1996.

[67] N. Wiberg, H.-A. Loeliger, and R. Kötter. Codes and iterative decoding on general graphs. *Euro. Trans. Telecom.*, 6:513–525, Sept./Oct. 1995.

[68] F.-Y. Wu. The Potts model. *Reviews of modern physics*, 54(1):235–268, 1982.

[69] V. A. Zinoviev and T. Ericson. Fourier-invariant pairs of partitions of finite abelian groups and association schemes. *Problems of Information Transmission*, 45(3):221–231, 2009.