A NEW LOOK AT THE GENERALIZED DISTRIBUTIVE LAW

by

Payam Pakzad, Venkat Anantharam

Memorandum No. UCB/ERL M01/32

10 June 2001

A NEW LOOK AT THE GENERALIZED DISTRIBUTIVE LAW

by

Payam Pakzad and Venkat Anantharam

Memorandum No. UCB/ERL M01/32

10 June 2001

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A New Look at the Generalized Distributive Law*

Payam Pakzad (payamp@eecs.berkeley.edu)
Venkat Anantharam (ananth@eecs.berkeley.edu)

June 2001
Berkeley, California

Abstract

In this paper we develop a measure-theoretic version of the Junction Tree algorithm to compute the marginalizations of a product function. We reformulate the problem in a measure-theoretic framework, where the desired marginalizations are viewed as conditional expectations of a product function given certain σ-fields. We generalize the notions of independence and junction trees at the level of these σ-fields and produce algorithms to find or construct a junction tree on a given set of σ–fields. By taking advantage of structures at the *atomic* level of the sample space, our marginalization algorithm is capable of producing solutions far less complex than GDL-type algorithms (see [1]). Our formalism reduces to the old GDL as a special case, and any GDL marginalization problem can be reexamined with our framework for possible savings in complexity.

1 Introduction

Local message-passing algorithms on graphs have enjoyed much attention in the recent years, mainly due to their success in decoding applications. A general framework for these algorithms was introduced by Shafer and Shenoy in [10]. This general framework is widely known as the Junction Tree algorithm in the Artificial Intelligence community. Aji and McEliece [1] gave an equivalent, (and for our purposes, slightly more convenient) framework known as the Generalized Distributive Law (GDL). Viterbi algorithm, BCJR [3], Belief Propagation [9], FFT over a finite field, and Turbo [7] and LDPC [6] decoding algorithms are amongst implementations of the Junction Tree algorithm or GDL[1].

Given a marginalization problem, GDL makes use of the *distributivity* of the 'product' operation over 'summation' in an underlying *semiring* to reduce the complexity of the required calculations. In many cases this translates to substantial savings, but as we shall see in this paper, sometimes there is just more

*This work was supported by grants (ONR/MURI) N00014-1-0637, (NSF) ECS-9873086 and (EPRI/DOD) EPRI-W08333-04

[1]Throughout this paper we will use both names – *GDL*, and the *Junction Tree algorithm*– interchangeably, but will use the GDL notation from [1] to show connections with this work.

structure in the local functions than GDL can efficiently handle. In particular, GDL relies solely on the notion of *variables*. Any structure at a finer level than that of variables will be ignored by GDL. We illustrate these limitations in the following simple example:

Example 1. Let X and Y be arbitrary real functions on $\{1, \cdots, n\}$. Let $\mu(i,j)$ be a fixed real weight function for $i,j \in \{1, \cdots, n\}$, given by an $n \times n$ matrix M with $\mu(i,j) = M_{i,j}$. We would like to calculate the weighted average of $X \cdot Y$: $E = \sum_{i=1}^{n} \sum_{j=1}^{n} X(i)Y(j)\mu(i,j)$.

The general GDL-type algorithm (assuming no structure on the weight function μ) will suggest the following:

$$E = \sum_{i=1}^{n} X(i) \sum_{j=1}^{n} Y(j)\mu(i,j),$$

requiring $n(n+1)$ multiplications and $(n-1)(n+1)$ additions. But this is not always the simplest way to calculate E.

Consider a 'luckiest' case when the matrix M has rank 1, i.e. the weight function $\mu(i,j)$ factorizes as $f_1(i) \cdot f_2(j)$. In this case $E = \left(\sum_{i=1}^{n} X(i)f_1(i)\right)\left(\sum_{j=1}^{n} Y(j)f_2(j)\right)$, requiring only $2n+1$ multiplications and $2n-2$ additions.

Suppose next that $\mu(i,j)$ does not factorize as above, but the matrix M has a low rank of 2, so that $\mu(i,j) = f_1(i)f_2(j) + g_1(i)g_2(j)$. Then we can compute E as follows:

$$E = \left(\sum_{i=1}^{n} X(i)f_1(i)\right)\left(\sum_{j=1}^{n} Y(j)f_2(j)\right) + \left(\sum_{i=1}^{n} X(i)g_1(i)\right)\left(\sum_{j=1}^{n} Y(j)g_2(j)\right)$$

This requires $4n+2$ multiplications and $4n-4$ additions.

Next suppose that the fixed weight matrix M is sparse, for example with $\mu(i,j) = m_i\delta(i+j-n-1)$. Then

$$E = \sum_{i=1}^{n} m_i X(i)Y(n+1-i),$$

requiring only $2n$ multiplications and $n-1$ additions.

From this example it is evident that there are families of marginalization problems for which the GDL treatment is insufficient to produce the best method of calculation.

In this paper we introduce a probability-theoretic framework which eliminates the explicit use of 'variables' to represent the states of the data. Specifically, we replace GDL's concept of 'local domains' with σ-fields in an appropriate sample-space. We also replace GDL's 'local kernels' with random variables measurable with respect to those σ-fields. The marginalization problem is then, naturally, replaced by taking the conditional expectation given a σ-field. As we shall see, this representation has the flexibility and natural tool (in form of

a measure function on the sample-space) to capture both full and partial independencies between the marginalizable functions. Our formalism reduces to the old GDL as a special case, and any GDL marginalization problem can be reexamined with our framework for possible savings in complexity.

Although our results are generalizable to any arbitrary *semifield*[2], in order to avoid abstract distractions, we focus on the sum-product algebra.

Here is an outline of this paper. In Section 2 we review the GDL algorithm. In Section 3 we present the necessary concepts from the probability theory. In Section 4 we give a probabilistic version of the marginalization problem that we address in this paper, and introduce the probabilistic junction trees and the message-passing algorithm on them. We further produce an algorithm to find a junction tree on a given collection of σ-fields. Just as is the case with the ordinary GDL, junction trees do not always exist. In Section 5 we discuss a method to expand the problem in a minimal way so to be able to construct a junction tree (much like the process of *moralization* and *triangulation*). Some examples and applications are given in Section 6, and in Section 7 we discuss our results.

2 GDL Algorithm

Definition 1. A (commutative) *semiring* R is a set with operations $+$ and $\times$ such that both $+$ and $\times$ are commutative and associative and have identity elements in R (0 and 1 respectively), and $\times$ is distributive over $+$.

Let $\{x_1, \cdots, x_n\}$ be variables taking values in sets $\{A_1, \cdots, A_n\}$ respectively. Let $\{S_1, \cdots, S_M\}$ be a collection of subsets of $\{1, \cdots, n\}$, and for $i \in \{1, \cdots, M\}$, let $\alpha_i : A_{S_i} \longrightarrow R$ be a function of x_{S_i}, taking value in some *semiring* R. The "Marginalize a Product Function" (MPF) problem is to find, for one or more of the indices $i = 1, \cdots, M$, the S_i-marginalization of the product of the α_i's, i.e.

$$\beta_i(x_{S_i}) \triangleq \sum_{x_{S_i^c} \in A_{S_i^c}} \Big(\prod_{i=1}^{M} \alpha_i(x_{S_i}) \Big)$$

In the language of GDL, α_i's are called the *local kernels*, and the variable lists x_{S_i} are called the *local domains*.

The GDL algorithm gives a message passing solution when the sets $\{S_1, \cdots, S_M\}$ can be organized into a *junction tree*. A junction tree is a graph-theoretic tree with nodes corresponding to $\{S_1, \cdots, S_M\}$, and with the property that the subgraph on the nodes that contain any variable x_i is connected. An equivalent

[2] A semifield is an algebraic structure with addition and multiplication, both of which are commutative and associative and have identity element. Further, multiplication is distributive over addition, and every nonzero element has a multiplicative inverse. Such useful algebras as the sum-product and the max-sum are examples of semifields (see [2]).

condition is that if A, B and C are subsets of $\{1, \cdots, M\}$ such that A and B are separated by S on the graph, then $S_A \cap S_B \in S_C$ where $S_A \triangleq \bigcup_{i \in A} S_i$. As we will see in Section 4 our definition of a junction tree will resemble this latter definition.

Suppose $\mathbf{G}$ is a junction tree on nodes $\{1, \cdots, M\}$ with local kernels $\{\alpha_1, \cdots, \alpha_M\}$. Let $\{E_1, \cdots, E_T\}$ be a message-passing *schedule*, viz. the 'message' function along the (directed) edge (i, j) of the graph is updated at time t iff $(i, j) \in E_t$. The following asynchronous message-passing algorithm (GDL) will solve the MPF problem:

Algorithm 1. *At each time t and for all pairs (i, j) of neighboring nodes in the graph let the 'message' from i to j be a function $\mu_{i,j}^t : A_{S_i \cap S_j} \longrightarrow R$. Initialize all 'message' functions to* 1. *At each time $t \in \{1, \cdots, T\}$, if the edge $(i, j) \in E_t$ then update the message from node i to j as follows*

$$\mu_{i,j}^t(x_{S_i \cap S_j}) = \sum_{x_{S_i \setminus S_j} \in A_{S_i \setminus S_j}} \alpha_i(x_{S_i}) \prod_{\substack{k \text{ adj } i \\ k \neq j}} \mu_{k,i}^{t-1}(x_{S_k \cap S_i}) \tag{1}$$

where (k adj i) means that node k is adjacent to i on the tree.

This algorithm will converge in finite time, at which time we have:

$$\alpha_i(x_{S_i}) \prod_{k \text{ adj } i} \mu_{k,i}(x_{S_k \cap S_i}) = \beta_i(x_{S_i}) = \sum_{x_{S_i^c} \in A_{S_i^c}} \Big(\prod_{i=1}^{M} \alpha_i(x_{S_i}) \Big)$$

Proof. See [1]. □

3 Probabilistic Preliminaries

First we review some notions from probability theory. Throughout this paper we focus on discrete sample spaces, i.e. the case when the sample space is finite or countably infinite.

Let $(\Omega, \mathcal{M})$ be a *discrete measurable space*, i.e. Ω is a finite or countable set and $\mathcal{M}$ is a σ-field on Ω. Let $\mu : \mathcal{M} \to (-\infty, \infty)$ be a *signed measure* on $(\Omega, \mathcal{M})$, i.e. $\mu(\emptyset) = 0$ and for any sequence $\{A_i\}_{i=1}^{\infty}$ of disjoint sets in $\mathcal{M}$, $\mu(\bigcup_1^\infty A_i) = \sum_1^\infty \mu(A_i)$. (As a matter of notation, we usually write $\mu(A_1, A_2, \cdots, A_n)$ for $\mu(A_1 \cap A_2 \cap \cdots \cap A_n)$.) Then we call $(\Omega, \mathcal{M}, \mu)$ a *measure space*. If $(\Omega, \mathcal{M}, \mu)$ is a measure space and $\{\mathcal{F}_1, \cdots, \mathcal{F}_M\}$ are sub σ-fields of $\mathcal{M}$, then we call $(\Omega, \{\mathcal{F}_1, \cdots, \mathcal{F}_M\}, \mu)$ a *collection of measure spaces.*

Let $\mathcal{F}, \mathcal{G}$ and $\mathcal{H}$ be sub σ-fields of $\mathcal{M}$.

Atoms of a σ-field: We define the set of *atoms* of $\mathcal{F}$ to be the collection of the minimal nonempty measurable sets in $\mathcal{F}$ w.r.t. inclusion:

$$\mathcal{A}(\mathcal{F}) \triangleq \{f \in \mathcal{F} \,:\, f \neq \emptyset, \text{ and } \forall g \in \mathcal{F},\ f \cap g \in \{\emptyset, f\}\}$$

Augmentation of σ-fields: We denote by $\mathcal{F} \vee \mathcal{G}$ the span of $\mathcal{F}$ and $\mathcal{G}$, i.e. the smallest σ-field containing both $\mathcal{F}$ and $\mathcal{G}$. For a set A of indices, we write $\mathcal{F}_A$ for $\bigvee_{i \in A} \mathcal{F}_i$, with $\mathcal{F}_\emptyset \triangleq \{\emptyset, \Omega\}$, the trivial σ-field on Ω. Note that the atoms of $\mathcal{F} \vee \mathcal{G}$ are all in the form $f \cap g$ for some $f \in \mathcal{A}(\mathcal{F})$ and $g \in \mathcal{A}(\mathcal{G})$.

Conditional Independence: We say $\mathcal{F}$ is conditionally independent of $\mathcal{G}$ given $\mathcal{H}$ or $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \,|\, \mathcal{H}$ w.r.t. μ when for any *atom* h of $\mathcal{H}$,

- if $\mu(h) = 0$ then $\forall f \in \mathcal{F}, g \in \mathcal{G}, \quad \mu(f, g, h) = 0$
- if $\mu(h) \neq 0$ then $\forall f \in \mathcal{F}, g \in \mathcal{G}, \quad \mu(f, g, h)\mu(h) = \mu(f, h)\mu(g, h)$.

When the underlying measure is obvious from the context, we omit the explicit mention of it.

Independence: We say $\mathcal{F}$ is independent of $\mathcal{G}$ or $\mathcal{F} \perp\!\!\!\perp \mathcal{G}$ w.r.t. μ when $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \,|\, \{\emptyset, \Omega\}$.

Note that these definitions are consistent with the usual definitions of independence when μ is a probability function.

Conditional Measure: Although it is not essential for our discussion, we define the conditional measure as a partially defined function $\mu(\cdot|\cdot) \,:\, \mathcal{M} \times \mathcal{M} \to (-\infty, \infty)$, defined as:

$$\mu(a|b) \triangleq \frac{\mu(a, b)}{\mu(b)} \quad \text{defined for nonzero-measure } b$$

Expectation and Conditional Expectation: Let $\mathcal{F}$ and $\mathcal{G}$ be a σ-fields with atoms $\mathcal{A}(\mathcal{F}) = \{f_i\}_{i=1}^{\infty}$ and $\mathcal{A}(\mathcal{G}) = \{g_i\}_{i=1}^{\infty}$ respectively. A *partially-defined* random variable X in $\mathcal{F}$ is a partially-defined function on Ω, where for each r in the range of X, $X^{-1}(r)$ is measurable in $\mathcal{F}$. We write $X \in \mathcal{F}$, and denote by $\mathcal{A}_X(\mathcal{F})$ the subset of $\mathcal{A}(\mathcal{F}) = \{f_i\}_{i=1}^{\infty}$ where X is defined.
Assuming $\mu(\Omega) \neq 0$, the *expectation* of X is defined as

$$\begin{aligned} \mathbf{E}[X] &\triangleq \frac{1}{\mu(\Omega)} \sum_{f \in \mathcal{A}_X(\mathcal{F})} X(f)\mu(f) \\ &= \sum_{f \in \mathcal{A}_X(\mathcal{F})} X(f)\mu(f|\Omega) \end{aligned}$$

Then we define the *conditional expectation* of X given $\mathcal{G}$, as a partially-defined random variable Y in $\mathcal{G}$, with $\mathcal{A}_Y(\mathcal{G}) = \{g \in \mathcal{A}(\mathcal{G}) \,:\, \mu(g) \neq 0\}$, as

follows:

$$\begin{aligned}\mathbf{E}[X|\mathcal{G}](g) &\triangleq \frac{1}{\mu(g)} \sum_{f \in \mathcal{A}_X(\mathcal{F})} X(f)\mu(g,f) && \text{for each atom } g \in \mathcal{A}_Y(\mathcal{G}) \\ &= \sum_{f \in \mathcal{A}_X(\mathcal{F})} X(f)\mu(f|g) && \text{for } g \in \mathcal{A}_Y(\mathcal{G})\end{aligned}$$

The signed Conditional Independence relation satisfies certain properties (inference rules) that we will state in the next theorem. See [9] and [5] for discussion of inference rules for the case when μ is a probability function[3].

Theorem 3.1. *Let $\mathcal{F}, \mathcal{G}, \mathcal{X}, \mathcal{Y}$ be σ-fields. Then the following properties hold:*

$$\mathcal{F} \perp\!\!\!\perp \mathcal{G} \,|\, \mathcal{X} \implies \mathcal{G} \perp\!\!\!\perp \mathcal{F} \,|\, \mathcal{X} \qquad \textit{Symmetry} \quad (2a)$$

$$\mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{X} \,|\, \mathcal{Y} \implies \mathcal{F} \perp\!\!\!\perp \mathcal{G} \,|\, \mathcal{Y} \;\;\&\;\; \mathcal{F} \perp\!\!\!\perp \mathcal{X} \,|\, \mathcal{Y} \qquad \textit{Decomposition} \quad (2b)$$

$$\mathcal{F} \perp\!\!\!\perp \mathcal{G} \,|\, \mathcal{X} \;\&\; \mathcal{F} \perp\!\!\!\perp \mathcal{Y} \,|\, \mathcal{G} \vee \mathcal{X} \implies \mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{Y} \,|\, \mathcal{X} \qquad \textit{Contraction} \quad (2c)$$

$$\mathcal{F} \perp\!\!\!\perp \mathcal{G} \,|\, \mathcal{X} \;\&\; \mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \,|\, \mathcal{X} \implies \mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{Y} \,|\, \mathcal{X} \qquad (2d)$$

$$\mathcal{F} \perp\!\!\!\perp \mathcal{G} \,|\, \mathcal{X} \;\&\; \mathcal{F} \vee \mathcal{X} \perp\!\!\!\perp \mathcal{Y} \,|\, \mathcal{G} \implies \mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{Y} \,|\, \mathcal{X} \qquad (2e)$$

$$\mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{X} \,|\, \mathcal{Y} \;\&\; \mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \,|\, \mathcal{X} \implies \begin{cases} \mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{Y} \,|\, \mathcal{X} \\ \mathcal{F} \vee \mathcal{Y} \perp\!\!\!\perp \mathcal{G} \,|\, \mathcal{X} \end{cases} \qquad (2f)$$

Proof. Let f, g, x, and y be arbitrary atoms of $\mathcal{F}, \mathcal{G}, \mathcal{X}$ and $\mathcal{Y}$ respectively. The proofs below consider all possible cases for the value of the μ on these atoms.

(2a): Symmetry is obvious, since $f \cap g = g \cap f$.

(2b): If $\mu(y) = 0$, then $\mu(f,g,x,y) = 0$ and if $\mu(y) \neq 0$, then $\mu(f,g,x,y) = \mu(f,y)\mu(g,x,y)/\mu(y)$ for all choices of f, g and x in $\mathcal{F}, \mathcal{G}$ and $\mathcal{X}$ respectively. In particular, choosing $x = \Omega$ or $g = \Omega$ will yield the desired results.

(2c):

- $\mu(x) = 0$. Then from $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \,|\, \mathcal{X}$, we get $\mu(g,x) = 0$ for all g. Then from $\mathcal{F} \perp\!\!\!\perp \mathcal{Y} \,|\, \mathcal{G} \vee \mathcal{X}$ we get $\mu(f,g,x,y) = 0$ for all f and y, and so we are done.
- $\mu(x) \neq 0$ and $\mu(g,x) = 0$. Then from $\mathcal{F} \perp\!\!\!\perp \mathcal{Y} \,|\, \mathcal{G} \vee \mathcal{X}$ we get $\mu(f,g,x,y) = 0$ for all f and y, and so $\mu(f,g,x,y)\mu(x) = \mu(f,x)\mu(g,x,y) = 0$ and we are done.

[3]Note that the signed Conditional Independence relation satisfies *symmetry, decomposition* and *contraction*, but in general *weak union* does not hold. So the signed C.I. relation is not a *semi-graphoid*.

- $\mu(x) \neq 0$ and $\mu(g,x) \neq 0$.Then from $\mathcal{F} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{G} \vee \mathcal{X}$ we get

$$\mu(f,g,x,y) = \mu(f,g,x)\mu(g,x,y)/\mu(g,x) \tag{3}$$

Also from $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \mid \mathcal{X}$ we have $\mu(f,g,x)/\mu(g,x) = \mu(f,x)/\mu(x)$. Replacing this into (3) we obtain $\mu(f,g,x,y) = \mu(g,x,y)\mu(f,x)/\mu(x)$ and we are done.

(2d):

- $\mu(x) = 0$. Then from $\mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{X}$, we get $\mu(f,g,x,y) = 0$ for all f, g and y, and so we are done.
- $\mu(x) \neq 0$ and $\mu(x,y) = 0$. Then from $\mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{X}$ we get $\mu(f,g,x,y) = \mu(f,g,x)\mu(x,y)/\mu(x) = 0$ for all f, g and y, and in particular $\mu(g,x,y) = 0$ and so we have the desired equality $\mu(f,g,x,y)/\mu(x) = \mu(f,x)\mu(g,x,y) = 0$.
- $\mu(x) \neq 0$ and $\mu(x,y) \neq 0$. We have

$$\mu(f,g,x) = \mu(f,x)\mu(g,x)/\mu(x) \qquad \text{since } \mathcal{F} \perp\!\!\!\perp \mathcal{G} \mid \mathcal{X} \tag{4}$$

$$\mu(f,g,x,y) = \mu(f,g,x)\mu(x,y)/\mu(x) \qquad \text{since } \mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{X} \tag{5}$$

$$\mu(g,x)/\mu(x) = \mu(g,x,y)/\mu(x,y) \qquad \text{since, by (2b), } \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{X} \tag{6}$$

Replacing (6) into (4) and then into (5) we obtain

$$\mu(f,g,x,y) = \mu(f,x)\mu(g,x,y)/\mu(x).$$

(2e):

- $\mu(x) = 0$ and $\mu(g) = 0$. Then from $\mathcal{F} \vee \mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{G}$, we get $\mu(f,g,x,y) = 0$ and we are done.
- $\mu(x) = 0$ and $\mu(g) \neq 0$. From $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \mid \mathcal{X}$ we have $\mu(f,g,x) = 0$. Then from $\mathcal{F} \vee \mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{G}$, $\mu(f,g,x,y) = \mu(f,g,x)\mu(y,g)/\mu(g) = 0$ and we are done.
- $\mu(x) \neq 0$ and $\mu(g) = 0$. Then from $\mathcal{F} \vee \mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{G}$, we get both $\mu(f,g,x,y) = 0$ and $\mu(g,x,y) = 0$, so the desired equality hold:

$$\mu(f,g,x,y) = \mu(f,x)\mu(g,x,y)/\mu(x) = 0.$$

- $\mu(x) \neq 0$ and $\mu(g) \neq 0$. Then from $\mathcal{F} \vee \mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{G}$, we get $\mu(f,g,x,y) = \mu(f,g,x)\mu(g,y)/\mu(g)$. Also from $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \mid \mathcal{X}$ we have

$$\mu(f,g,x) = \mu(f,x)\mu(g,x)/\mu(x).$$

So we obtain the equality $\mu(f,g,x,y) = \mu(f,x)\mu(g,x)\mu(g,y)/(\mu(g)\mu(x))$. Finally, *decomposition* applied to $\mathcal{F} \vee \mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{G}$ yields $\mu(g,x)\mu(g,y)/\mu(g) = \mu(g,x,y)$. So we have proved $\mu(f,g,x,y) = \mu(f,x)\mu(g,x,y)/\mu(x)$ and this completes the proof.

(2f):

- $\mu(x) = 0$. Then from $\mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{X}$, we have $\mu(f,g,x,y) = 0$ and we are done.
- $\mu(x) \neq 0$ and $\mu(x,y) = 0$. Then from $\mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{X}$ we have $\mu(f,g,x,y) = \mu(f,g,x)\mu(x,y)/\mu(x)$ and so $\mu(f,g,x,y) = 0$. Also after applying (2b) to the above, we have $\mu(f,x,y) = \mu(f,x)\mu(x,y)/\mu(x) = 0$ and $\mu(g,x,y) = \mu(g,x)\mu(x,y)/\mu(x) = 0$. So we have the equality $\mu(f,g,x,y)\mu(x) = \mu(f,x,y)\mu(g,x) = \mu(f,x)\mu(g,x,y) = 0$ and we are done.
- $\mu(x) \neq 0$ and $\mu(x,y) \neq 0$. Then also $\mu(y) \neq 0$ or else from $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{X} \mid \mathcal{Y}$ we would have $\mu(x,y) = 0$. Then from $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{X} \mid \mathcal{Y}$ we get $\mu(f,g,x,y) = \mu(f,y)\mu(g,x,y)/\mu(y)$, and also after (2b) to the above, we get $\mu(f,x,y)/\mu(x,y) = \mu(f,y)/\mu(y)$. Replacing the latter equation into the former we obtain

$$\mu(f,g,x,y) = \mu(g,x,y)\mu(f,x,y)/\mu(x,y) \tag{7}$$

  But from $\mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{X}$ and by (2b) we have both $\mu(f,x,y)/\mu(x,y) = \mu(f,x)/\mu(x)$ and $\mu(g,x,y)/\mu(x,y) = \mu(g,x)/\mu(x)$. Replacing each of these into (7) we obtain

$$\mu(f,g,x,y) = \mu(g,x,y)\mu(f,x)/\mu(x)$$

  and

$$\mu(f,g,x,y) = \mu(f,x,y)\mu(g,x)/\mu(x)$$

  and we are done.

□

4 Probabilistic MPF and Junction Trees

We now formulate a probabilistic version of the MPF problem and introduce the corresponding concept of *junction trees*. The rest of this paper will analyze properties of these junction trees and describe a probabilistic GDL algorithm to solve this MPF problem.

Throughout this paper, let $(\Omega, \{\mathcal{F}_1, \cdots, \mathcal{F}_M\}, \mu)$ be a collection of measure spaces, and let $\{X_1, \cdots, X_M\}$ be a collection of partially defined random variables with $X_i \in \mathcal{F}_i$.

Probabilistic MPF Problem: For one or more $i \in \{1, \cdots, M\}$, find $\mathbf{E}[\prod_j X_j | \mathcal{F}_i]$, the conditional expectation of the product given $\mathcal{F}_i$.

A GDL MPF problem in the format described in Section 2 can be represented as a probabilistic MPF problem usually in more than one way, depending on

the choice of assignment of GDL's *local kernels* as either a random variable, or a factor of the measure function; either way, the product will be the same. Specifically, a marginalization $\sum_{x_{S_i^c} \in A_{S_i^c}} \left(\prod_{i=1}^{M} \alpha_i(x_{S_i}) \right)$ can be viewed as a weighted average of the product of α_j's for $j \in J \subset \{1, \cdots, M\}$ with the measure function $\mu(x_{\{1,\cdots,M\}}) = \prod_{k \in J^c} \alpha_k(x_{S_k})$ for any subset J of $\{1, \cdots, M\}$. Our sample space Ω is then the product space $A_{\{1,\cdots,M\}} = A_1 \times \cdots \times A_M$. For each $j \in J$, we view α_j as a random variable measurable in a σ-field whose atoms are the hyper-planar subsets of Ω in which the coordinates corresponding to the elements of S_j are constant. In other words, each atom of this σ-field corresponds to a possible choice of $x_{S_j} \in A_{S_j}$. Denoting each atom by its corresponding element x_{S_j} then, we have

$$\mathbf{E}[\prod_i X_i | \mathcal{F}_j](x_{S_j}) = \frac{1}{\mu(x_{S_j})} \sum_{x_{S_i^c} \in A_{S_i^c}} \left(\prod_{i=1}^{M} \alpha_i(x_{S_i}) \right) = \frac{1}{q_{S_j^c}} \beta_j(x_{S_j})$$

In most applications, for a family of MPF problems the local kernels can be categorized as either *fixed* or *arbitrary*. For example, in an LDPC decoding problem, the code itself is fixed, so the local kernels at the check-nodes are fixed; we only receive new observations and try to find the most likely codeword given each observation set. As another example, when finding the Hadamard transform $\sum_{x_1,\cdots,x_n} \prod_{i=1}^{n} (-1)^{x_i y_i} f(x_1, \cdots, x_n)$ of an arbitrary function f, the functions $(-1)^{x_i y_i}$ are fixed. Typically, we want to assign (some of) the fixed kernels as the measure function, and the arbitrary kernels as the marginalizable random variables; this way, once a junction tree has been found for one problem, it can be used to marginalize the product of any arbitrary collection of random variables measurable in the same σ-fields. See Section 6 for more examples.

4.1 Junction Trees

As in the case of the old GDL, junction trees are defined to capture the underlying independencies in the marginalizable functions. Given the above problem setup we define junction trees as follows:

Definition 2. Let G be a tree with nodes $\{1, \cdots, M\}$. We say subsets A and B of $\{1, \cdots, M\}$ are *separated* by a node i if $\forall x \in A, y \in B$, the path from x to y contains i. Then we call G a *Junction Tree* if $\forall A, B \subset \{1, \cdots, M\}$ and $i \in \{1, \cdots, M\}$ s.t. i separates A and B on the tree we have $\mathcal{F}_A \perp\!\!\!\perp \mathcal{F}_B \mid \mathcal{F}_i$.

Lemma 4.1. *Suppose there exists a junction tree with nodes corresponding to σ-fields $\{\mathcal{F}_1, \cdots, \mathcal{F}_M\}$. Then if f is a zero measure atom of any of the $\mathcal{F}_i$'s, and $g \subset f$ is measurable in $\bigvee_{i=1}^{M} \mathcal{F}_i$, then $\mu(g) = 0$.*

Proof. Node i vacuously separates the empty subset of $\{1, \cdots, M\}$ from $\{1, \cdots, M\} \backslash \{i\}$. Thus $\{\emptyset, \Omega\} \perp\!\!\!\perp \bigvee_{\substack{j=1 \\ j \neq i}}^{M} \mathcal{F}_j \mid \mathcal{F}_i$. Thus by the definition of conditional independence, whenever $f \in \mathcal{A}(\mathcal{F}_i)$ has zero measure, all its subsets measurable in $\bigvee_{i=1}^{M} \mathcal{F}_i$ also have measure zero. □

Lemma 4.2. *Let $\mathcal{F}_1, \mathcal{F}_2$ and $\mathcal{F}_3$ be σ-fields such that $\mathcal{F}_1 \perp\!\!\!\perp \mathcal{F}_3 \mid \mathcal{F}_2$. Then for any partially-defined random variable $X \in \mathcal{F}_1$, the following equality holds:*

$$\mathbf{E}\Big[\mathbf{E}\big[X\big|\mathcal{F}_2\big]\Big|\mathcal{F}_3\Big] = \mathbf{E}\big[X\big|\mathcal{F}_3\big]$$

Proof. Let $Y = \mathbf{E}\big[X\big|\mathcal{F}_2\big]$. Then $A_Y(\mathcal{F}_2) = \{b \in \mathcal{A}(\mathcal{F}_2) \ : \ \mu(b) \neq 0\}$ and for $b \in A_Y(\mathcal{F}_2)$, $Y(b) = \frac{1}{\mu(b)} \sum_{a \in A_X(\mathcal{F}_1)} X(a)\mu(a,b)$. Then, for any nonzero-measure atom $c \in \mathcal{A}(\mathcal{F}_3)$,

$$\begin{aligned}
\mathbf{E}\Big[\mathbf{E}\big[X\big|\mathcal{F}_2\big]\Big|\mathcal{F}_3\Big](c) &= \mathbf{E}\big[Y\big|\mathcal{F}_3\big](c) \\
&= \frac{1}{\mu(c)} \sum_{b \in A_Y(\mathcal{F}_2)} Y(b)\mu(b,c) \\
&= \frac{1}{\mu(c)} \sum_{b \in A_Y(\mathcal{F}_2)} \frac{1}{\mu(b)} \sum_{a \in A_X(\mathcal{F}_1)} X(a)\mu(a,b)\mu(b,c) \\
&= \frac{1}{\mu(c)} \sum_{b \in A_Y(\mathcal{F}_2)} \sum_{a \in A_X(\mathcal{F}_1)} X(a)\mu(a,b,c) \qquad \text{since } \mathcal{F}_1 \perp\!\!\!\perp \mathcal{F}_3 \mid \mathcal{F}_2 \\
&= \frac{1}{\mu(c)} \sum_{a \in A_X(\mathcal{F}_1)} X(a) \sum_{b \in A_Y(\mathcal{F}_2)} \mu(a,b,c) \\
&= \frac{1}{\mu(c)} \sum_{a \in A_X(\mathcal{F}_1)} X(a)\mu(a,c) \\
&= \mathbf{E}\big[X\big|\mathcal{F}_3\big](c)
\end{aligned}$$

where we have used the fact that $\mathcal{F}_1 \perp\!\!\!\perp \mathcal{F}_3 \mid \mathcal{F}_2$, so $\sum_{b \in A_{\mathcal{F}_2}(Y)} \mu(a,b,c) = \sum_{b \in \mathcal{A}(\mathcal{F}_2)} \mu(a,b,c) = \mu(a,c)$. □

Lemma 4.3. *Let $\{\mathcal{F}_1, \cdots, \mathcal{F}_l\}$ and $\mathcal{F}$ be σ-fields such that $\{\mathcal{F}_1, \cdots, \mathcal{F}_l\}$ are mutually conditionally independent given $\mathcal{F}$. For each $i = 1, \cdots, l$, let X_i be a partially-defined random variable in $\mathcal{F}_i$. Then:*

$$\mathbf{E}\Big[\prod_{i=1}^{l} X_i \Big| \mathcal{F}\Big] = \prod_{i=1}^{l} \mathbf{E}\big[X_i\big|\mathcal{F}\big]$$

Proof. We shall proceed by induction. The statement is vacuous for $l = 1$. For $l = 2$, let $Y = \mathbf{E}\big[X_1X_2\big|\mathcal{F}\big]$. Then $A_Y(\mathcal{F}) = \{f \in \mathcal{A}(\mathcal{F}) \ : \ \mu(f) \neq 0\}$. Also note that X_1X_2 is a partially-defined random variable in $\mathcal{F}_1 \vee \mathcal{F}_2$ with $A_{X_1X_2}(\mathcal{F}_1 \vee \mathcal{F}_2) = A_{X_1}(\mathcal{F}_1 \vee \mathcal{F}_2) \cap A_{X_2}(\mathcal{F}_1 \vee \mathcal{F}_2)$, and that any atom of $\mathcal{F}_1 \vee \mathcal{F}_2$ can be written as $a \cap b$ for $a \in \mathcal{A}(\mathcal{F}_1)$ and $b \in \mathcal{A}(\mathcal{F}_2)$. Then for any $f \in A_Y(\mathcal{F})$ we have:

$$
\begin{aligned}
Y(f) &= \frac{1}{\mu(f)} \sum_{\substack{(a\cap b)\in \mathcal{A}_{X_1X_2}(\mathcal{F}_1\vee\mathcal{F}_2)\\ a\in\mathcal{F}_1, b\in\mathcal{F}_2}} X_1(a)X_2(b)\mu(a,b,f) \\
&= \frac{1}{\mu(f)} \sum_{a\in\mathcal{A}_{X_1}(\mathcal{F}_1)} \sum_{b\in\mathcal{A}_{X_2}(\mathcal{F}_2)} X_1(a)X_2(b)\frac{\mu(a,f)\mu(b,f)}{\mu(f)} \\
&= \frac{1}{\mu(f)} \sum_{a\in\mathcal{A}_{X_1}(\mathcal{F}_1)} X_1(a)\mu(a,f)\frac{1}{\mu(f)} \sum_{b\in\mathcal{A}_{X_2}(\mathcal{F}_2)} X_2(b)\mu(b,f) \\
&= \mathbf{E}[X_1|\mathcal{F}](f)\mathbf{E}[X_2|\mathcal{F}](f)
\end{aligned}
$$

where we have used the fact that $\mathcal{F}_1 \perp\!\!\!\perp \mathcal{F}_2 \mid \mathcal{F}$, so $\frac{\mu(a,f)\mu(b,f)}{\mu(f)} = \mu(a,b,f)$.

For $l > 2$ assume inductively that the equality holds for $l-1$. Then:

$$
\begin{aligned}
\mathbf{E}\Big[\prod_{i=1}^{l} X_i \Big| \mathcal{F}\Big] &= \mathbf{E}\Big[X_1 \prod_{i=2}^{l} X_i \Big| \mathcal{F}\Big] \\
&= \mathbf{E}[X_1|\mathcal{F}]\mathbf{E}\Big[\prod_{i=2}^{l} X_i \Big| \mathcal{F}\Big] \qquad \text{since } \mathcal{F}_1 \perp\!\!\!\perp \bigvee_{i=2}^{l} \mathcal{F}_i \mid \mathcal{F} \\
&= \prod_{i=1}^{l} \mathbf{E}[X_i|\mathcal{F}] \qquad \text{by induction hypothesis}
\end{aligned}
$$

□

4.2 Probabilistic Junction Tree Algorithm

Suppose G is a junction tree with σ–fields $\{\mathcal{F}_1, \cdots, \mathcal{F}_M\}$, as defined above, and let $\{X_1, \cdots, X_M\}$ be random variables with $X_i \in \mathcal{F}_i$. Then the following asynchronous message-passing algorithm will solve the probabilistic MPF problem:

Algorithm 2. *For each edge (i,j) on the graph, define a 'message' $Y_{i,j}$ from node i to j as a partially-defined random variable measurable in $\mathcal{F}_j$.*
For each $i = 1, \cdots, M$ define the set of neighbors of i as:

$$N_i = \{k \,:\, (i,k) \text{ an edge in the tree}\},$$

and for each edge (i,j) in the tree, define:

$$N_{i,j} = N_i \backslash \{j\}$$

Initialize all the messages to 1. *For each edge (i,j) in the tree, update the message $Y_{i,j}$ (asynchronously) as:*

$$Y_{i,j} = \mathbf{E}\Big[X_i \prod_{k\in N_{i,j}} Y_{k,i} \Big| \mathcal{F}_j\Big] \tag{8}$$

This algorithm will converge in finite time, at which time we have:

$$\mathbf{E}\Big[\prod_{j=1}^{M} X_j \Big| \mathcal{F}_i\Big] = X_i \prod_{k \in N_i} Y_{k,i}$$

Proof. The scheduling theorem 3.1 in [1] also holds here, using Lemmas 4.2 and 4.3 above. For completeness we will include that proof here.

We will show that if E_t is the schedule for activation of the nodes, (i.e. a directed edge $(i,j) \in E_t$ iff node i updates its message to its neighbor, j at time t) then the message from a node i to a neighboring node j is:

$$Y_{i,j}(t) = \mathbf{E}\Big[\prod_{k \in K_{i,j}(t)} X_k \Big| \mathcal{F}_j\Big], \tag{9}$$

where $K_{i,j}(t)$ is a subset of the nodes defined recursively by:

$$K_{i,j}(t) = \begin{cases} \emptyset & \text{if } t = 0, \\ K_{i,j}(t-1) & \text{if } (i,j) \notin E_t, \\ \{i\} \bigcup_{l \in N_{i,j}} K_{l,i}(t-1) & \text{if } (i,j) \in E_t \end{cases}$$

We will prove this by induction on t. Case $t = 0$ is clear from the initialization. Now let $t > 0$ and assume that (9) above holds for $t-1$. We can also assume that the $(i,j) \in E_t$ so the message $Y_{i,j}$ is being updated at time t. Then:

$$\begin{aligned}
Y_{i,j}(t) &= \mathbf{E}\Big[X_i \prod_{l \in N_{i,j}} Y_{k,i} \Big| \mathcal{F}_j\Big] \\
&= \mathbf{E}\Big[X_i \prod_{l \in N_{i,j}} \mathbf{E}\Big[\prod_{k \in K_{l,i}(t-1)} X_k | \mathcal{F}_i\Big] \Big| \mathcal{F}_j\Big] && \text{by induction} \\
&= \mathbf{E}\Big[\mathbf{E}\Big[X_i \prod_{l \in N_{i,j}} \prod_{k \in K_{l,i}(t-1)} X_k | \mathcal{F}_i\Big] \Big| \mathcal{F}_j\Big] && \text{by J.T. property and Lemma 4.3} \\
&= \mathbf{E}\Big[X_i \prod_{l \in N_{i,j}} \prod_{k \in K_{l,i}(t-1)} X_k \Big| \mathcal{F}_j\Big] && \text{by J.T. property and Lemma 4.2} \\
&= \mathbf{E}\Big[\prod_{k \in K_{i,j}(t)} X_k \Big| \mathcal{F}_j\Big] && \text{by definition of } K_{i,j}(t)
\end{aligned}$$

Indeed $K_{i,j}(t)$ above is the set of all the nodes whose 'information' has reached the edge (i,j) by time t. Similarly, with $J_i(t) \triangleq \{i\} \bigcup_{j \in N_i} K_{j,i}(t)$, $J_i(t)$ is the collection of all the nodes whose 'information' has reached a node i by time t. It is natural to think of a *message trellis* up to time t, which is an $M \times t$ directed graph, where for any $i,j \in \{1, \cdots, M\}$ and $n < t$, $i(n)$ is always connected to $i(n+1)$, and $i(n)$ is connected to $j(n+1)$ iff $(i,j) \in E_n$. It follows that we will have $J_i(t) = \{1, \cdots, M\}$ when there is a path from every the initial

node (i.e. at $t = 0$) in the trellis to the node $i(t)$. Then, since the tree has finite diameter, *any* infinite schedule that activates all the edges infinitely many times has a finite sub-schedule, say of length t_0 such that $J_i(t_0) = \{1, \cdots, M\}$ for all i. At that time we have:

$$\begin{aligned}
\mathbf{E}\Big[X_i \prod_{j\in N_i} Y_{j,i}(t_0)\Big|\mathcal{F}_i\Big] &= \mathbf{E}\Big[X_i \prod_{j\in N_i} \mathbf{E}\Big[\prod_{k\in K_{j,i}(t_0)} X_k\Big|\mathcal{F}_i\Big]\Big|\mathcal{F}_i\Big] && \text{by (9)}\\
&= \mathbf{E}\Big[\mathbf{E}\Big[X_i \prod_{j\in N_i}\prod_{k\in K_{j,i}(t_0)} X_k\Big|\mathcal{F}_i\Big]\Big|\mathcal{F}_i\Big] && \text{by J.T. property and Lemma 4.3}\\
&= \mathbf{E}\Big[X_i \prod_{j\in N_i}\prod_{k\in K_{j,i}(t_0)} X_k\Big|\mathcal{F}_i\Big]\\
&= \mathbf{E}\Big[\prod_{k\in J_i(t_0)} X_k\Big|\mathcal{F}_i\Big] && \text{by defn. of } J_i(t)\\
&= \mathbf{E}\Big[\prod_{k=1}^{M} X_k\Big|\mathcal{F}_i\Big]
\end{aligned}$$

□

4.3 Representation and Complexity

In this section we discuss the complexity of representation of a junction tree and the implementation of our algorithm.

Let $\mathbf{G}$ be a junction tree with σ-fields $\{\mathcal{F}_1, \cdots, \mathcal{F}_M\}$, as defined above, and let $\{X_1, \cdots, X_M\}$ be arbitrary random variables with $X_i \in \mathcal{F}_i$. Denote by q_i the number of atoms of the σ-field $\mathcal{F}_i$, so $q_i = |\mathcal{A}(\mathcal{F}_i)|$.

It can be seen that, in general, the sample space Ω can have as many as $\prod_{i=1}^{M} q_i$ elements and thus full representation of σ-fields and the measure function requires exponentially large storage resources. Fortunately, however, a full representation is not required. Along each edge (i,j) on the tree, Algorithm 2 only requires local computation of $\mathbf{E}[X|\mathcal{F}_i]$ for a random variable $X \in \mathcal{F}_j$. This only requires a $q_i \times q_j$ table of the joint measures of the atoms of $\mathcal{F}_i$ and $\mathcal{F}_j$. For an arbitrary edge (i,j), let $\mathcal{A}(\mathcal{F}_i) = \{a_1, \cdots, a_{q_i}\}$ and $\mathcal{A}(\mathcal{F}_j) = \{b_1, \cdots, b_{q_j}\}$ be the sets of atoms of $\mathcal{F}_i$ and $\mathcal{F}_j$. Define $W(i,j)$ to be the $q_i \times q_j$ matrix with (r,s) entry equal to $\mu(a_r|b_s)$; note that from Lemma 4.1, (possibly after trivial simplification of the problem by eliminating the zero measure events,) no atom of $\mathcal{F}_j$ can have measure 0, so $\mu(a_r|b_s)$ is defined for all atoms of $\mathcal{F}_j$. Then once a junction tree has been found, we need only keep $2(M-1)$ such matrices (corresponding to the $(M-1)$ edges of the tree) to fully represent the algorithm, for a total of $2\sum_{(i,j)\text{ an edge}} q_i q_j$ storage units.

The arithmetic complexity of the algorithm depends on even a smaller quantity, namely the total number of nonzero elements of the $W(i,j)$ matrices. Let $nz(i,j)$ denote the number of nonzero entries of the matrix $W(i,j)$ (note that

$nz(i,j) = nz(j,i)$). Let X be an arbitrary random variable in $\mathcal{F}_i$. Then

$$\begin{aligned}\mathbf{E}[X|\mathcal{F}_j](b_s) &= \sum_{r=1}^{q_i} X(a_r)\mu(a_r|b_s) \\ &= \sum_{r:\mu(a_r|b_s)\neq 0} X(a_r)\mu(a_r|b_s)\end{aligned}$$

requiring $nz(i,j)$ multiplications and $nz(i,j) - q_j$ additions. Note that the measures are assumed to be fixed, and only the X_i's are allowed to be arbitrary random variables measurable in the $\mathcal{F}_i$'s. So it makes sense to exclude multiplications and additions by the 0's from the algorithm.

For each (directed) edge (i,j) in $\mathbf{G}$ define $\chi(i,j) = 2nz(i,j) - q_j$ to be the *edge complexity*, i.e. the number of additions and multiplications required for computing $\mathbf{E}[X_i|\mathcal{F}_j]$. From the Algorithm 2, calculating the conditional expectation given a single σ-field $\mathcal{F}_i$ with the most efficient schedule, requires updating of the messages from the leaves towards the node i. Each edge is activated in one direction, and at each non-leaf node l the messages need to be multiplied to update the message from l to its neighbor in the direction of i. This requires, for each edge (k,l), an additional q_l multiplications. Thus the grand total arithmetic operations needed to calculate $\mathbf{E}[\prod_j X_j|\mathcal{F}_i]$ is $\sum_{(k,l) \text{ an edge}} 2nz(k,l)$.

Note that $nz(k,l)$ can be upper-bounded by $q_k q_l$, corresponding to carrying out multiplications and additions for the events of measure zero.

The complexity of the full algorithm, in which $\mathbf{E}[\prod_j X_j|\mathcal{F}_i]$ is calculated for all $i = 1, \cdots, M$, can also be found using similar ideas. For each node k, let $d(k)$ denote the number of the neighbors of k on the tree. Then for each directed edge (k,l), the $d(k) - 1$ messages from other neighbors of k must be multiplied by X_k (requiring $(d(k) - 1)q_k$ multiplications) and then the conditional expectation given $\mathcal{F}_l$ be taken (requiring $\chi(k,l)$ operations). So the total number of operations required for the full algorithm is

$$\begin{aligned}&\sum_{(k,l) \text{ a dir. edge}} 2nz(k,l) - q_l + (d(k) - 1)q_k \\ &= \sum_{(k,l) \text{ a dir. edge}} \left(2nz(k,l) + d(k)q_k\right) - \sum_{k=1}^{M} 2d(k)q_k \\ &= \sum_{(k,l) \text{ an edge}} 4nz(k,l) + \sum_{k=1}^{M}(d(k)^2 - 2d(k))q_k\end{aligned}$$

As noted in [1], it is possible to produce all the products of $d(k) - 1$ of $d(k)$ messages going into node k in a more efficient way. In this case, the total arithmetic complexity of the complete algorithm will be $\sum_{(k,l) \text{ an edge}} 4nz(k,l) + O(\sum_{k=1}^{M} d(k)q_k)$.

4.4 Existence of Junction Trees

Definition 3. A *Valid Partition* of $\{1,\cdots,M\}\backslash\{i\}$ with respect to a node i is a partition $\{p_1,\cdots,p_l\}$ of $\{1,\cdots,M\}\backslash\{i\}$ (i.e. $\bigcup_{j=1}^{M} p_j = \{1,\cdots,M\}\backslash\{i\}$ and $p_i \cap p_j = \emptyset$ for $i \neq j$) such that $\mathcal{F}_{p_j}$'s are mutually conditionally independent, given $\mathcal{F}_i$.

Definition 4. Let $P = \{p_1,\cdots,p_l\}$ be any partition of $\{1,\cdots,M\}\backslash\{i\}$. A tree with nodes $\{1,\cdots,M\}$ is called *compatible* with partition P at node i if its subtrees hanging from i correspond to the elements of P.

Lemma 4.4. *$\forall i \in \{1,\cdots,M\}$, there is a* Finest Valid Partition *w.r.t. i, which we shall denote by P_i, such that every other valid partition w.r.t. i is a coarsening of P_i. Further, if p is an element of P_i and p is the disjoint union of nonempty sets e_1 and e_2, then $\mathcal{F}_{e_1} \not\perp\!\!\!\perp \mathcal{F}_{e_2} \mid \mathcal{F}_i$.*

Proof. Suppose $A = \{p_1,\cdots,p_l\}$ and $B = \{q_1,\cdots,q_m\}$ are valid partitions w.r.t. node i. Now construct another partition, $C = \{p \cap q : p \in A \;\&\; q \in B\}$. We claim that C is also a valid partition w.r.t. i, (finer than both A and B): To see this, we need to show that for each $d = p \cap q \in C$, $\mathcal{F}_d \perp\!\!\!\perp \mathcal{F}_{d^c} \mid \mathcal{F}_i$. Using simple manipulations like $\mathcal{F}_p = \mathcal{F}_{p\cap(q\cup q^c)} = \mathcal{F}_{p\cap q} \vee \mathcal{F}_{p\cap q^c}$ we get:

$$\mathcal{F}_{p\cap q} \vee \mathcal{F}_{p\cap q^c} \perp\!\!\!\perp \mathcal{F}_{p^c} \mid \mathcal{F}_i$$
$$\mathcal{F}_{p\cap q} \vee \mathcal{F}_{p^c\cap q} \perp\!\!\!\perp \mathcal{F}_{p\cap q^c} \vee \mathcal{F}_{p^c\cap q^c} \mid \mathcal{F}_i \Rightarrow \mathcal{F}_{p\cap q} \perp\!\!\!\perp \mathcal{F}_{p\cap q^c} \mid \mathcal{F}_i \qquad \text{by (2b)}$$

And finally, the last two relations and (2d) imply that $\mathcal{F}_{p\cap q} \perp\!\!\!\perp \mathcal{F}_{p^c\cup(p\cap q^c)} \mid \mathcal{F}_i$, and hence $\mathcal{F}_{p\cap q} \perp\!\!\!\perp \mathcal{F}_{(p\cap q)^c} \mid \mathcal{F}_i$. So a finest valid partition w.r.t. i exists, whose atoms are the intersections of atoms of all the valid partitions w.r.t. i.

Now suppose p is an element of P_i and p is the disjoint union of nonempty sets e_1 and e_2, and $\mathcal{F}_{e_1} \perp\!\!\!\perp \mathcal{F}_{e_2} \mid \mathcal{F}_i$. We also have $\mathcal{F}_{e_1} \vee \mathcal{F}_{e_2} \perp\!\!\!\perp \mathcal{F}_{p^c} \mid \mathcal{F}_i$. Then from the last two relations and by (2d) we get $\mathcal{F}_{e_1} \perp\!\!\!\perp \mathcal{F}_{p^c} \vee \mathcal{F}_{e_2} \mid \mathcal{F}_i$, and hence $\mathcal{F}_{e_1} \perp\!\!\!\perp \mathcal{F}_{e_1^c} \mid \mathcal{F}_i$. Then e_1 and e_2 would be elements in a finer valid partition which is a contradiction. □

Lemma 4.5. *Given $\{\mathcal{F}_1,\cdots,\mathcal{F}_M\}$, a tree with nodes $\{1,\cdots,M\}$ is a junction tree iff at each node i it is compatible with some valid partition of $\{1,\cdots,M\}\backslash\{i\}$ w.r.t. i.*

Proof. Immediate from the definitions. □

Lemma 4.6. *Let d be a subset of $\{1,\cdots,M\}$ and let d' be its complement in $\{1,\cdots,M\}$. Suppose there exist $t \in d$ and $i \in d'$ such that $\mathcal{F}_d \perp\!\!\!\perp \mathcal{F}_{d'} \mid \mathcal{F}_t$ and $\mathcal{F}_d \perp\!\!\!\perp \mathcal{F}_{d'} \mid \mathcal{F}_i$. Let $\mathbf{G}$ be any junction tree on d and $\mathbf{G}'$ any junction tree on d'. Then the tree obtained by connecting $\mathbf{G}$ and $\mathbf{G}'$ by adding an edge between t and i is a junction tree on $\{1,\cdots,M\}$ (see Figure 1.)*

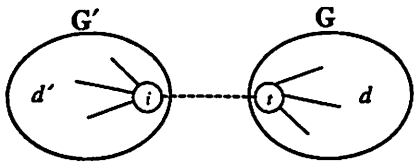

Figure 1: Augmenting Junction Trees; Lemma 4.6

Proof. Let x be any node that separates A and B on the resultant tree. We will show that $\mathcal{F}_A \perp\!\!\!\perp \mathcal{F}_B \mid \mathcal{F}_x$ and hence we have a junction tree.
Let $A_1 = A \cap d, A_2 = A \cap d', B_1 = B \cap d$ and $B_2 = B \cap d'$ and WLOG suppose $x \in d$. Then at least one of A_2 and B_2 must be empty, or else x would not separate A and B. Suppose $A_2 = \emptyset$.

First suppose $x = t$. Then we have:

$$\mathcal{F}_{A_1} \perp\!\!\!\perp \mathcal{F}_{B_1} \mid \mathcal{F}_t \qquad \text{by J.T. property on } \mathbf{G}$$
$$\mathcal{F}_{A_1} \vee \mathcal{F}_{B_1} \perp\!\!\!\perp \mathcal{F}_{B_2} \mid \mathcal{F}_t \qquad \text{since } A_1 \cup B_1 \subset d \text{ and } B_2 \subset d'$$

So by (2d) we have $\mathcal{F}_{A_1} \perp\!\!\!\perp \mathcal{F}_{B_1} \vee \mathcal{F}_{B_2} \mid \mathcal{F}_t$, i.e. $\mathcal{F}_A \perp\!\!\!\perp \mathcal{F}_B \mid \mathcal{F}_t$ and we are done.
Next Suppose $x \in d \backslash \{t\}$. Then we must also have that x separates A_1 from $B_1 \cup \{t\}$ (assuming that B_2 is nonempty, which is no loss of generality.) Then:

$$\mathcal{F}_{A_1} \perp\!\!\!\perp \mathcal{F}_{B_1} \vee \mathcal{F}_t \mid \mathcal{F}_x \tag{10}$$
$$\mathcal{F}_{A_1} \vee \mathcal{F}_x \vee \mathcal{F}_{B_1} \perp\!\!\!\perp \mathcal{F}_{B_2} \mid \mathcal{F}_t \qquad \text{since } A_1 \cup B_1 \cup \{x\} \subset d \text{ and } B_2 \subset d' \tag{11}$$

We will show that $\mathcal{F}_{A_1} \perp\!\!\!\perp \mathcal{F}_{B_1} \vee \mathcal{F}_{B_2} \vee \mathcal{F}_t \mid \mathcal{F}_x$ and hence $\mathcal{F}_A \perp\!\!\!\perp \mathcal{F}_B \mid \mathcal{F}_x$.
Let $\chi, \tau, \alpha, \beta_1$ and β_2 be arbitrary atoms of $\mathcal{F}_x, \mathcal{F}_t, \mathcal{F}_A, \mathcal{F}_{B_1}$ and $\mathcal{F}_{B_2}$ respectively.

- *Case* $\mu(\chi) = \mu(\tau) = 0$. Then from (11) we have that $\mu(\alpha, \beta_1, \beta_2, \chi, \tau) = 0$, and so we are done.

- *Case* $\mu(\chi) = 0$ *and* $\mu(\tau) \neq 0$. Then from (11) we have $\mu(\alpha, \beta_1, \beta_2, \chi, \tau) = \mu(\alpha, \beta_1, \chi, \tau)\mu(\beta_2, \tau)/\mu(\tau)$. But from (10), $\mu(\alpha, \beta_1, \chi, \tau) = 0$ since $\mu(\chi) = 0$. Thus $\mu(\alpha, \beta_1, \beta_2, \chi, \tau) = 0$ and we are done.

- *Case* $\mu(\chi) \neq 0$ *and* $\mu(\tau) = 0$. Then from (11) we have that $\mu(\alpha, \beta_1, \beta_2, \chi, \tau) = \mu(\beta_1, \beta_2, \chi, \tau) = 0$, and so we have the equality $\mu(\alpha, \beta_1, \beta_2, \chi, \tau)\mu(\chi) = \mu(\alpha, \chi)\mu(\beta_1, \beta_2, \chi, \tau) = 0$ and we are done.

- *Case* $\mu(\chi) \neq 0$ *and* $\mu(\tau) \neq 0$. Then from (11),

$$\mu(\alpha, \beta_1, \beta_2, \chi, \tau) = \mu(\alpha, \beta_1, \chi, \tau)\mu(\beta_2, \tau)/\mu(\tau),$$

and from (10), $\mu(\alpha,\beta_1,\chi,\tau) = \mu(\alpha,\chi)\mu(\beta_1,\chi,\tau)/\mu(\chi)$. Replacing the latter into the former, we obtain

$$\mu(\alpha,\beta_1,\beta_2,\chi,\tau) = \mu(\alpha,\chi)\mu(\beta_1,\chi,\tau)\mu(\beta_2,\tau)/(\mu(\chi)\mu(\tau)).$$

But by (11), $\mu(\beta_1,\chi,\tau)\mu(\beta_2,\tau)/\mu(\tau) = \mu(\beta_1,\beta_2,\chi,\tau)$, so

$$\mu(\alpha,\beta_1,\beta_2,\chi,\tau) = \mu(\alpha,\chi)\mu(\beta_1,\beta_2,\chi,\tau)/\mu(\chi)$$

and we are done.

□

We now state our main theorem on the existence of junction trees:

Theorem 4.7. *Given a set of σ-fields $\{\mathcal{F}_1,\cdots,\mathcal{F}_M\}$, if there exists a junction tree on $\{1,\cdots,M\}$, then for every $i \in \{1,\cdots,M\}$ there exists a junction tree compatible with P_i, the finest valid partition w.r.t. i.*

Notice that Theorem 4.7 and Lemma 4.6 effectively give an algorithm to find a junction tree, when one exists, as we shall describe in Section 4.5.

Proof. The claim is trivial for $M \leq 3$. We will prove the theorem for $M > 3$ by induction:
Let $P_i = \{c_1,\cdots,c_l\}$ with $\bigcup_{j=1}^{l} c_j = \{1,\cdots,M\}\backslash\{i\}$ and $c_j \cap c_k = \emptyset$ for $j \neq k$. Let $\mathbf{G}$ be a junction tree. Let $Q = \{d_1,\cdots,d_n\}$ be the partition of $\{1,\cdots,M\}\backslash\{i\}$ compatible with $\mathbf{G}$. Let $d = d_j$ be an arbitrary element of Q, and let $d' = \bigcup_{k\neq j} d_k \cup \{i\}$. Let $t = N_i \cup d$ be the node in d that is neighbor to i in tree $\mathbf{G}$. By Lemmas 4.4 and 4.5 above, d is the union of some of c_k's. WLOG assume that $d = \bigcup_{k=1}^{K} c_k$ where $K \leq l$, and also assume that $t \in c_K$.
Then from the junction tree property, we have

$$\mathcal{F}_i \perp\!\!\!\perp \mathcal{F}_d \mid \mathcal{F}_t \tag{12}$$

Since $\mathbf{G}$ is a junction tree, the subtree on d is also a junction tree. Now $|d| < M$, and so by induction hypothesis there exists a junction tree on d compatible with P'_t, the finest valid partition w.r.t. t of $d\backslash\{t\}$.
Now we claim that $R = \{c_k\backslash\{t\} \ : \ 1 \leq k \leq K\}$ is a valid partition of $d\backslash\{t\}$ w.r.t. t. To see this, let $c = c_k$ for some arbitrary $k = 1,\cdots,K$, and let $c' = d\backslash\{c\}$, so $\mathcal{F}_d = \mathcal{F}_c \vee \mathcal{F}_{c'}$. But one of c and c' contains t. Then by the properties of valid partition w.r.t. i, we have:

$$\mathcal{F}_c \vee \mathcal{F}_t \perp\!\!\!\perp \mathcal{F}_{c'} \mid \mathcal{F}_i \qquad \text{or} \qquad \mathcal{F}_c \perp\!\!\!\perp \mathcal{F}_{c'} \vee \mathcal{F}_t \mid \mathcal{F}_i$$

$$\text{also,} \qquad \mathcal{F}_i \perp\!\!\!\perp \mathcal{F}_c \vee \mathcal{F}_{c'} \mid \mathcal{F}_t \quad \text{since } t \text{ separates } i \text{ from } d \text{ on } \mathbf{G}$$

Then by (2f) followed by (2b), the last relations imply that $\mathcal{F}_c \perp\!\!\!\perp \mathcal{F}_{c'} \mid \mathcal{F}_t$ and we are done.

Next we show that for all $k \in \{1, \cdots, K-1\}$ (so that $t \notin c_k$), c_k is an element of P_t'. If not, then there exists a $c = c_k \in R$, with $t \notin c$, s.t. c is the disjoint union of some subsets e_1 and e_2 and $\mathcal{F}_{e_1} \perp\!\!\!\perp \mathcal{F}_{e_2} \mid \mathcal{F}_t$. Also $\mathcal{F}_{e_1} \vee \mathcal{F}_{e_2} \perp\!\!\!\perp \mathcal{F}_i \mid \mathcal{F}_t$ so by (2d) we get $\mathcal{F}_{e_1} \perp\!\!\!\perp \mathcal{F}_{e_2} \vee \mathcal{F}_t \mid \mathcal{F}_i$. We also have $\mathcal{F}_{e_1} \vee \mathcal{F}_{e_2} \perp\!\!\!\perp \mathcal{F}_t \mid \mathcal{F}_i$ since $e_1 \cup e_2 = c$ and t belongs to another set in the finest valid partition w.r.t. i. From the last two relations and by (2f) followed by (2b) we get $\mathcal{F}_{e_1} \perp\!\!\!\perp \mathcal{F}_{e_2} \mid \mathcal{F}_i$. But by Lemma 4.4, $c \in P_i$ cannot be so decomposed, so $\{e_1, e_2\} = \{c, \emptyset\}$ and we have proved the claim.

So we have shown, by induction, that there exists a junction tree, $\mathbf{G}_d$ on d, where node t has at least $K-1$ neighbors with subtrees corresponding to c_k, $1 \leq k \leq K-1$. Now we modify the original junction tree, $\mathbf{G}$ in $K+1$ steps to get trees $\mathbf{H}$, $\mathbf{H}_0, \cdots, \mathbf{H}_{K-1}$ as follows:

First we form $\mathbf{H}$ by replacing the subtree in $\mathbf{G}$ on d, with $\mathbf{G}_d$ above, connecting i to t with an edge. By Lemma 4.6, $\mathbf{H}$ is a junction tree on $\{1, \cdots, M\}$.

Let $\mathbf{H}_0$ be the subtree of $\mathbf{H}$ after removing the subtrees around t on c_k, $1 \leq k \leq K-1$. Then $\mathbf{H}_0$ is also a junction tree. For each $j = 1, \cdots, K-1$ let $\mathbf{L}_j$ be the subtree of $\mathbf{H}$ on c_j, and let x_j be the node on c_j that was connected to t in $\mathbf{H}$. Then at each step $j = 1, \cdots, K-1$ we form $\mathbf{H}_j$ by joining $\mathbf{H}_{j-1}$ and $\mathbf{L}_j$ by adding the edge between i and x_j (see Figure 2.)

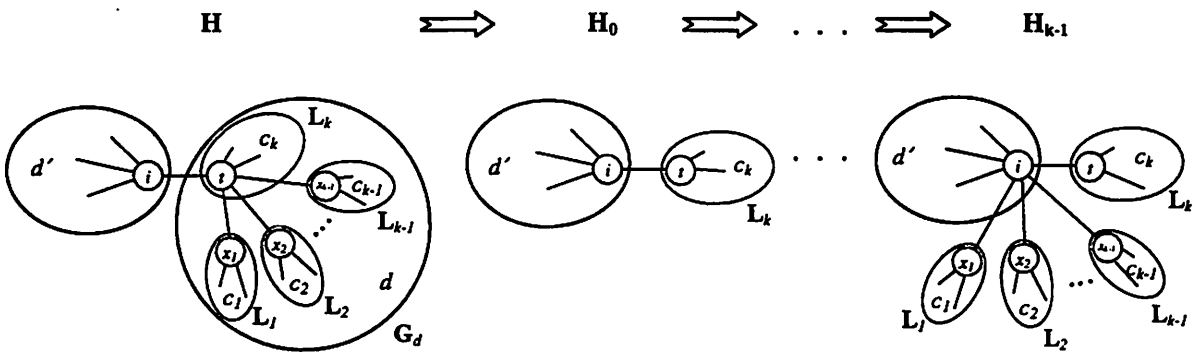

Figure 2: Transformation of Junction Trees

We now show inductively that each $\mathbf{H}_j$ is a junction tree. By induction hypothesis $\mathbf{H}_{j-1}$ is a junction tree. At the same time, $\mathbf{L}_j$, being a subtree of a junction tree, is also a junction tree. Further $F_{c_j} \perp\!\!\!\perp \mathcal{F}_{c_K} \vee \mathcal{F}_{d'} \bigvee_{r=1}^{j-1} c_r \mid \mathcal{F}_i$, since c_j is a set in a valid partition w.r.t. i.
Also, $F_{c_j} \perp\!\!\!\perp \mathcal{F}_{c_K} \vee \mathcal{F}_{d'} \bigvee_{r=1}^{j-1} c_r \mid \mathcal{F}_{x_j}$, since on the junction tree $\mathbf{H}$, node x_j separates c_j from $c_K \cup d' \bigcup_{r=1}^{j-1} c_r$. Then by Lemma 4.6, each $\mathbf{H}_j$ is a junction tree (Note that $\mathbf{H}_{K-1}$ is a junction tree on $\{1, \cdots, M\}$.)

Next we perform the same transformation on $\mathbf{H}_{K-1}$, starting with other neighbors of i. The resulting tree will be a junction tree, and will be compatible with P_i. □

4.5 Algorithm to Find a Junction Tree

We will now produce an algorithm to find a junction tree when one exists. Given a set of σ-fields $\{\mathcal{F}_1, \cdots, \mathcal{F}_M\}$,

Algorithm 3. *Pick any node $i \in \{1, \cdots, M\}$ as the root.*

- *If $M = 2$ then the single edge $(1, 2)$ is a junction tree. Stop.*
- *Find the finest valid partition of $\{1, \cdots, M\} \backslash \{i\}$ w.r.t. i, $P_i = \{c_1, \cdots, c_l\}$ (see notes below).*
- *For j=1 to l*
- *Find a node $t \in c_j$ s.t. $\mathcal{F}_i \perp\!\!\!\perp \mathcal{F}_{c_j} \mid \mathcal{F}_t$. If no such node exists, then stop; no junction tree exists.*
- *Find a junction tree on c_j with node t as root. Attach this tree, by adding edge (i, t).*
- *End For*

Note: In the general case of the signed conditional independence, we know of no better way to find the finest valid partition than an exhaustive search in an exponential subset of all the partitions. In the case of unsigned measures, however, we can show that when a junction tree exists, the finest valid partition coincides with the finest pairwise partition, which can be found in polynomial time.

Proof. At each iteration, t is chosen so $\mathcal{F}_i \perp\!\!\!\perp \mathcal{F}_{c_j} \mid \mathcal{F}_t$. But we also had $\mathcal{F}_{c_j^c} \perp\!\!\!\perp \mathcal{F}_t \vee \mathcal{F}_{c_j} \mid \mathcal{F}_i$. By (2e) the last two relations imply $\mathcal{F}_{c_j} \perp\!\!\!\perp \mathcal{F}_i \vee \mathcal{F}_{c_j^c} \mid \mathcal{F}_t$. But we also have $\mathcal{F}_{c_j} \perp\!\!\!\perp \mathcal{F}_i \vee \mathcal{F}_{c_j^c} \mid \mathcal{F}_i$. So by Lemma 4.6 we have a junction tree at each step. Also, from Theorem 4.7 if the algorithm fails, then there is no junction tree. □

5 Construction of Junction Tree - Lifting

In the previous section we gave an algorithm to find a junction tree, when one exists. In this section we deal with the case when Algorithm 3 declares that no junction tree exists for the given set of σ-fields. In particular, we would like to modify the σ-fields in some *minimal* sense, so as to ensure that we can construct a junction tree.

Definition 5. Let $(\Omega, \{\mathcal{F}_1, \cdots, \mathcal{F}_M\}, \mu)$ be a collection of measure spaces. We call another collection of measure spaces $(\Omega', \{\mathcal{F}'_1, \cdots, \mathcal{F}'_{M'}\}, \mu')$ a *lifting* of $(\Omega, \{\mathcal{F}_1, \cdots, \mathcal{F}_M\}, \mu)$ if there are maps, $f : \Omega' \longrightarrow \Omega$ and $\sigma : \{1, \cdots, M\} \longrightarrow \{1, \cdots, M\}$ such that:

- μ' is consistent with μ under the map f, i.e.
  $\forall\, A \in \mathcal{F}_{\{1,\cdots,M\}}, \quad \mu(A) = \mu'(f^{-1}(A))$.

- For all $i = 1, \cdots, M$, f is $(\mathcal{F}'_{\sigma(i)}, \mathcal{F}_i)$-*measurable*, i.e. $\forall\, A \in \mathcal{F}_i,\ f^{-1}(A) \in \mathcal{F}'_{\sigma(i)}$.

where for $A \in \Omega$, $f^{-1}(A) \triangleq \{\omega' \in \Omega' \,:\, f(\omega') \in A\}$.

In words, up to some renaming of the elements, each σ-field $\mathcal{F}_i$ is a sub-σ-field of some $\mathcal{F}'_j$, (namely, $\mathcal{F}'_{\sigma(i)}$) and this $\mathcal{F}'_j$ is obtained from $\mathcal{F}_i$ by *splitting* some of the atoms.

We now describe the connection of the above concept with our problem.

Let $(\Omega', \{\mathcal{F}'_1, \cdots, \mathcal{F}'_{M'}\}, \mu')$ be a lifting of $(\Omega, \{\mathcal{F}_1, \cdots, \mathcal{F}_M\}, \mu)$ with lifting maps $f : \Omega' \longrightarrow \Omega$ and $\sigma : \{1, \cdots, M\} \longrightarrow \{1, \cdots, M\}$ as described in Definition 5. Let $\mathbf{G}'$ be a junction tree on $\{1, \cdots, M'\}$ corresponding to σ-fields $\{\mathcal{F}'_1, \cdots, \mathcal{F}'_{M'}\}$. We will construct a junction tree $\mathbf{G}''$ from $\mathbf{G}'$ such that the running Algorithm 2 on $\mathbf{G}''$ will produce the desired conditional expectations at appropriate nodes.

For each $i = 1, \cdots, M$, let $\mathcal{G}_i$ be the σ-field on Ω' with atoms $\mathcal{A}(\mathcal{G}_i) = \{f^{-1}(a) : a \in \mathcal{A}(\mathcal{F}_i)\}$, and let $Y_i \in \mathcal{G}_i$ be the random variable with $Y_i(f^{-1}(a)) = X_i(a)$ for all $a \in \mathcal{A}(\mathcal{F}_i)$; so that up to a renaming of the atoms and elements, $(\Omega, \mathcal{F}_i, \mu)$ and $(\Omega', \mathcal{G}_i, \mu')$ are the same measure space and X_i and Y_i are the same random variable. Let $\mathbf{G}''$ be a tree with nodes $\{1, \cdots, M', M'+1, \cdots, M'+M\}$, – with corresponding σ-fields $\{\mathcal{F}'_1, \cdots, \mathcal{F}'_{M'}, \mathcal{G}'_1, \cdots, \mathcal{G}'_M\}$ and random variables $\{1, \cdots, 1, Y_1, \cdots, Y_M\}$,– which is generated by starting with $\mathbf{G}'$ and adding edges $(j, M' + \sigma(j))$ for each $j = 1, \cdots, M'$. In words, $\mathbf{G}''$ is a graph obtained from $\mathbf{G}'$ by adding and attaching each node with σ-fields $\mathcal{G}_i$ for $i = 1, \cdots, M$ (which are in turn equivalent to the original $\mathcal{F}_i$'s,) to the node whose σ-field contains that $\mathcal{G}_i$. Then by Lemma 4.6, $\mathbf{G}''$ is a junction tree and hence running Algorithm 2 on $\mathbf{G}''$ will produce $\mathbf{E}\big[\prod_{i=1}^{M} Y_i \big| \mathcal{G}_j\big]$ at the node labelled $(M'+\sigma(j))$ for each $j = 1, \cdots, M$. But these are equivalent to $\mathbf{E}\big[\prod_{i=1}^{M} X_i \big| \mathcal{F}_j\big]$ for $j = 1, \cdots, M$ and we have thus solved the probabilistic MPF problem.

So we only need to establish how to lift a certain collection of σ-fields to create the required independencies and form a junction tree.

Suppose we have three σ-fields, $\mathcal{F}_1, \mathcal{F}_2$ and $\mathcal{F}_3$, and we would like to lift the measure space $(\Omega, \{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}, \mu)$ into $(\Omega', \{\mathcal{F}'_1, \mathcal{F}'_2, \mathcal{F}'_3\}, \mu')$ in a way to have the independence $\mathcal{F}'_1 \perp\!\!\!\perp \mathcal{F}'_3 \mid \mathcal{F}'_2$. Let $a_i \in \mathcal{A}(\mathcal{F}_1)$, $c_k \in \mathcal{A}(\mathcal{F}_2)$ and $b_j \in \mathcal{A}(\mathcal{F}_3)$ be arbitrary atoms. For each c_k, let A_k be the matrix with (i, j) entry equal to $\mu(a_i, b_j, c_k)$. Then $\mu(c_k) = 0$ iff the sum of entries of A_k is zero. However we cannot have a nonzero matrix with zero sum of entries. If this is the case, we can decompose the matrix into sum of two matrices with nonzero sum of entries. This corresponds to splitting atom c_k in a way that the new atoms have nonzero measure.

Next for each such matrix, A_k with nonzero sum of entries, the independence condition corresponding to c_k is exactly the condition that A_k is rank-one. If, however, A_k is *not* rank-one, we can split c_k using an *optimal* decomposition

of A_k as the sum of say q rank-one matrices so that none of the matrices are zero-sum.[4] This corresponds to splitting the atom c_k into q atoms, $\{c_1^k, \cdots, c_q^k\}$ where each of c_l^k's render $\mathcal{F}_1$ and $\mathcal{F}_3$ independent. c_l^k's are then atoms of $\mathcal{F}_2'$.

5.1 Algorithm to Construct a Junction Tree

Combining the above ideas with the Algorithm 3 we obtain:

Algorithm 4. *Pick any node $i \in \{1, \cdots, M\}$ as the root.*

- *If $M = 2$ then the single edge $(1,2)$ is a junction tree. Stop.*
- *Find any[5] valid partition of $\{1, \cdots, M\} \backslash \{i\}$ w.r.t. i, $P_i = \{c_1, \cdots, c_l\}$.*
- *For j=1 to l*
- *Find a node $t \in c_j$ s.t. $\mathcal{F}_i \perp\!\!\!\perp \mathcal{F}_{c_j} \mid \mathcal{F}_t$. If no such node exists, then pick any $t \in c_j$. Lift $\mathcal{F}_t$ by splitting some of its atoms as discussed above so to have $\mathcal{F}_i \perp\!\!\!\perp \mathcal{F}_{c_j} \mid \mathcal{F}_t$ (see notes below).*
- *Find a junction tree on c_j with node t as root. Attach this tree, by adding edge (i,t).*
- *End For*

The resulting collection $(\Omega', \{\mathcal{F}_1', \cdots, \mathcal{F}_{M'}'\}, \mu')$ is a lifting of the original collection with $M' = M$, and the tree generated by this algorithm is a junction tree corresponding to this lifted collection of σ-fields.

In many cases, however, it is possible to achieve a less complex junction tree algorithm by allowing M' to exceed M. The following optional steps can be used for possible reduction of complexity:

- *For each edge (i,j) in the resultant junction tree,*
- *Create a σ-field $\mathcal{G}_{(i,j)}$ that renders $\mathcal{F}_i$ and $\mathcal{F}_j$ independent, i.e. $\mathcal{F}_i \perp\!\!\!\perp \mathcal{F}_j \mid \mathcal{G}_{(i,j)}$. This can be done by starting with $\mathcal{F}_{\{\emptyset,\Omega\}}$ and applying the rank-one decomposition techniques in the previous section.*
- *Insert a new node in the tree corresponding to $\mathcal{G}_{(i,j)}$, between i and j.*
- *End For*

[4] This can always be done with $q = rank(A_k)$ if A_k is not zero-sum. Obviously $rank(A_k)$ is also a lower bound for q. An optimal decomposition, however, not only aims to minimize q, but also involves minimizing the number of nonzero entries of the decomposing matrices, as discussed in Section 4.3.

[5] Although any valid partition will work, in general finer partitions should result in better and less complex algorithms (see Section 4.3).

Notes: In general, the size of the Ω space can grow multiplicatively with each 'lifting,' so the full representation of the measure spaces require exponentially large storage resources. However, as mentioned in Section 4.3, a full description of the algorithm requires storing only the $W(i,j)$ matrices of the conditional measures of the atoms of the neighboring σ-fields. In fact, a careful reader may have noticed that we have not completely specified the lifting maps as defined in Definition 5 on the entire collection of measure spaces. In fact once the lifting has been done on a subset of the σ-fields (so to create the desired independencies), there is a unique way to update the measure function on the entire sample space that ensures that previous relations still hold. Such extension, however, is unnecessary since by the consistency of the measures in a lifting, the matrices of conditional measures along other edges of the tree remain the same.

6 Examples

In this section we consider a few examples where GDL-based algorithms are applicable. We will work out the details of the process of making a junction tree, and compare the complexity of the message-passing algorithm with GDL.

Our Example 1 at the beginning of this paper can be generalized as follows:

Example 2. Let A_i, $i = 1, \cdots, M$ be finite sets of size q, and for each $i = 1, \cdots, M$, let X_i be a real function on A_i. Let μ be a real (weight) function on $A_1 \times A_2 \times \cdots \times A_M$. We would like to compute the weighted average of the product of X_i's, i.e. $E = \sum_{a_i \in A_i} X_1(a_1)X_2(a_2)\cdots X_M(a_M)\mu(a_1, \cdots, a_M)$. Suppose that the weight function is in the following form:

$$\mu(a_1, \cdots, a_M) = \prod_{i=1}^{M} f_i(a_i) + \prod_{i=1}^{M} g_i(a_i)$$

As long as the weight function μ is not in product form, the most efficient GDL algorithm will prescribe

$$E = \sum_{a_1 \in A_1} X_1(a_1) \cdots \sum_{a_M \in A_M} X_M(a_M)\mu(a_1, \cdots, a_M)$$

requiring $O(n^M)$ additions and multiplications, corresponding to the following junction tree.

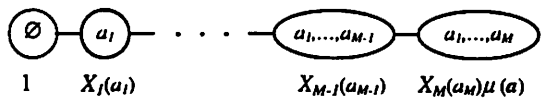

Figure 3: GDL Junction Tree

Now, Let Ω be the product space $A_1 \times A_2 \times \cdots \times A_M$ with signed measure μ, and for $i = 1, \cdots, M$, let $\mathcal{F}_i$ be the σ–field containing the planes $a_i = c$, so $X_i \in \mathcal{F}_i$. Let $\mathcal{F}_0 = \{\emptyset, \Omega\}$ be the trivial σ–field. Then the problem is to find the conditional expectation of the product of the X_i's given $\mathcal{F}_0$.
The best junction tree is obtained by lifting the space so that given $\mathcal{F}_0'$ all other σ–fields are mutually conditionally independent. To do this, we split the atom Ω of $\mathcal{F}_0$ into two atoms Ω_1 and Ω_2. In effect, for each element $(a_1, \cdots, a_M)$ of Ω, the new space Ω' has two elements $(a_1, \cdots, a_M) \cap \Omega_1$ and $(a_1, \cdots, a_M) \cap \Omega_2$. The new weight function is defined on Ω' as $\mu'((a_1, \cdots, a_M^1) \cap \Omega_1) = \prod_{i=1}^M f_i(a_i)$ and $\mu'((a_1, \cdots, a_M) \cap \Omega_1) = \prod_{i=1}^M g_i(a_i)$.

Then there is a star-shaped junction tree on $\{0, \cdots, M\}$ with node 0 at the center. The message passing algorithm on this tree is:

$$\begin{aligned} E &= \mathbf{E}[\prod X_i | \mathcal{F}_0](\Omega) \\ &= \mathbf{E}[\prod X_i | \mathcal{F}_0'](\Omega_1) + \mathbf{E}[\prod X_i | \mathcal{F}_0'](\Omega_2) \\ &= \prod_{i=1}^M \mathbf{E}[X_i | \mathcal{F}_0'](\Omega_1) + \prod_{i=1}^M \mathbf{E}[X_i | \mathcal{F}_0'](\Omega_2) \\ &= \prod_{i=1}^M \sum_{a_i \in A_i} X_i(a_i) f_i(a_i) + \prod_{i=1}^M \sum_{a_i \in A_i} X_i(a_i) g_i(a_i) \end{aligned}$$

Note that this requires only $O(Mn)$ additions and multiplications.

In the next example we show that Pearl's treatment of the belief propagation algorithm in the case of a node with *disjunctive interaction* or *noisy-or-gate* causation ([9], section 4.3.2) can be viewed as a special case of our algorithm:

Example 3. Bayesian Network with Disjunctive Interaction. Let the binary inputs $U = (U_1, U_2, \cdots, U_n)$ be the parents of the binary node X in the Bayesian network of Figure 4, interacting on X through a noisy-or-gate.

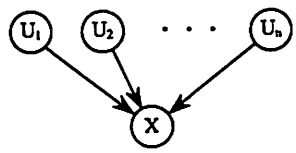

Figure 4: Bayesian Network of Example 3

This means that there are parameters $q_1, \cdots, q_n \in [0, 1]$ so that

$$P(X = 0 \mid U) = \prod_{i \in T_u} q_i$$

$$P(X = 1 \mid U) = 1 - \prod_{i \in T_u} q_i$$

where $T_u \triangleq \{i : U_i = 1\}$.

Normal moralization and triangulation technique applied to this graph will give a single clique with all the variables. However, because of the structure in the problem, a better solution exists.

Let $\mathcal{F}_X, \mathcal{F}_1, \cdots, \mathcal{F}_n$ be the σ-fields generated by the (independent) variables $x, u_1, \cdots, u_n$ respectively. All variables are binary so each of the σ-fields has precisely two atoms. In our framework, let the 'random variables' be (the function) $1 \in \mathcal{F}_X$, and $\pi_X(u_i) = P(U_i = u_i) \in \mathcal{F}_i$ for $i = 1, \cdots, n$, with the underlying joint measure on $x, u_1, \cdots, u_n$ defined to be $\mu(x, u_1, \cdots, u_n) = P(X = x \mid U = (u_1, \cdots, u_n))$. Then F_i's are *not* mutually conditionally independent given $\mathcal{F}_X$, however the following simple lifting of space will create the independence: Let a variable x' be defined to take value in $0, 1, 2$, where the event $\{x' = 0\}$ corresponds to $\{x = 0\}$, and $\{x' = 1\} \cup \{x' = 2\}$ correspond to $\{x = 1\}$. Extend the measure as follows:

$$\mu'(x', u_1, \cdots, u_n) = \begin{cases} \prod_{i \in T_u} q_i & \text{if } x' = 0 \\ -\prod_{i \in T_u} q_i & \text{if } x' = 1 \\ 1 & \text{if } x' = 2 \end{cases}$$

Then we see that in this *lifted* spaces, the σ-fields $\mathcal{F}'_i$ (generated by variables u_i respectively) are mutually conditionally independent given $\mathcal{F}'_{X'}$ (the σ-field generated by variable x'.) Then we have a junction tree in the shape of a star, with $\mathcal{F}'_{X'}$ corresponding to the central node. The junction tree algorithm will calculate the following marginalized random variable at the node corresponding to $\mathcal{F}'_{X'}$:

$$\beta(x') = \begin{cases} \prod_{i=1}^n \left(P(U_i = 0) + P(U_i = 1)\, q_i\right) & \text{if } x' = 0 \\ -\prod_{i=1}^n \left(P(U_i = 0) + P(U_i = 1)\, q_i\right) & \text{if } x' = 1 \\ 1 & \text{if } x' = 2 \end{cases}$$

Then the *belief* at X is the function

$$\mathrm{BEL}(x) = \begin{cases} \prod_{i=1}^n \left(P(U_i = 0) + P(U_i = 1)\, q_i\right) & \text{if } x = 0 \\ 1 - \prod_{i=1}^n \left(P(U_i = 0) + P(U_i = 1)\, q_i\right) & \text{if } x = 1 \end{cases}$$

where we have merged the atoms $x' = 1$ and $x' = 2$ of $\mathcal{F}'_{X'}$ to get back $x = 1$. This is essentially the same as Equation (4.57) in [9].

Example 4. Hadamard Transform. Let $x_1, \cdots, x_n$ be binary variables and let $f(x_1, \cdots, x_n)$ be a real function of the x's. The Hadamard transform of f is defined as

$$g(y_1, \cdots, y_n) = \sum_{x_1, \cdots, x_n} \prod_{i=1}^n (-1)^{x_i y_i} f(x_1, \cdots, x_n)$$

where $y_1, \cdots, y_n$ are binary variables.

Since our framework is particularly useful when the underlying functions are structured, we consider the case when f is a symmetric function of $x_1, \cdots, x_n$,

i.e. f depends only on the sum of the x_i's. Then it is easy to verify
Claim: When f is symmetric, then its Hadamard transform, g is also a symmetric function.

We now set up the problem in our framework. Let Ω be $\{0,1\}^{2n}$ with elements $\omega = (x_1, \cdots, x_n, y_1, \cdots, y_n)$. Let $\mathcal{F}$ and $\mathcal{G}$ be the σ–fields in which respectively f and g are measurable; in our case of symmetric f and g, $\mathcal{A}(\mathcal{F}) = \{\alpha_k \text{ for } k = 0, \cdots, n\} = \{\{\omega : \sum_i x_i = k\} \text{ for } k = 0, \cdots, n\}$ and $\mathcal{A}(\mathcal{G}) = \{\beta_k \text{ for } k = 0, \cdots, n\} = \{\{\omega : \sum_i y_i = k\} \text{ for } k = 0, \cdots, n\}$.
Next we note that all the factors involving terms $(-1)^{x_i y_i}$ can be summarized as a signed measure μ on $\mathcal{F} \vee \mathcal{G}$ as follows:

$$\begin{aligned}
\mu(\alpha_j, \beta_k) &= \sum_{\omega \in \alpha_j \cap \beta_k} (-1)^{\sum_i x_i y_i} \\
&= \sum_{\substack{\omega: \sum_i x_i = j, \\ \sum_i y_i = k}} (-1)^{\sum_i x_i y_i} \\
&= \sum_{(x_1, \cdots, x_n): \sum_i x_i = j} (-1)^{\sum_{i=1}^{k} x_i}
\end{aligned}$$

Note that μ can be stored in a $(n+1) \times (n+1)$ table.

Now we have a junction tree with only two nodes, corresponding to $\mathcal{F}$ and $\mathcal{G}$, and the marginalization is done as follows:

$$\begin{aligned}
g(\beta_k) &= \mathbf{E}[f|\mathcal{G}] \\
&= \sum_{j=0}^{n} f(\alpha_j) \mu(\alpha_j, \beta_k)
\end{aligned}$$

where $f(\alpha_j) = f\big((x_1, \cdots, x_n) : \sum_i x_i = j\big)$ and $g(\beta_k) = g\big((y_1, \cdots, y_n) : \sum_i y_i = k\big)$.

This requires only n additions and $(n+1)$ multiplications for each of $(n+1)$ possible values of g.

We have created a Matlab library containing the necessary functions to set up a general marginalization problem and create a junction tree. The following examples are processed using that code.

Example 5. Probabilistic State Machine. Consider the Bayesian network depicted in Figure 5, where u_i, s_i and y_i denote inputs, hidden states and the outputs of a chain of length n. Let m be the memory of the state machine, so that each state s_{i+1} can be taken to be $(u_{i-m+1}, \cdots, u_i)$.

The problem of finding the maximum-likelihood input symbols is solved by using the BCJR algorithm [3], which is an implementation of GDL algorithm on the junction tree obtained by moralization and triangulation of the above Bayesian network. The local functions are $P(s_0), P(u_i), P(y_i^c|u_i, s_i)$ and $P(s_i|u_{i-1}, s_{i-1})$. At each stage i, a message-update involves marginalizing a

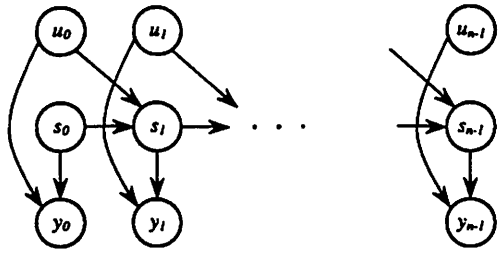

Figure 5: Bayesian Network for a Probabilistic State Machine

function $f(u_{i-m}, \cdots, u_i)$ over u_{i-m}. So in case of binary inputs and outputs, the BCJR algorithm will require about $(5\,n\,2^m)$ additions and multiplications.

Now consider a case when the output of the state machine depends on the input and state in a simple, but non-product form. For the purposes of this example, we have chosen the output y_i to be the outcome of an 'OR' gate on the state s_i and input u_i, passed through a binary symmetric channel, i.e.

$$P(y_i|u_i, s_i) = (1-p)1(y_i = \vee_{j=0}^{m} u_{i-j}) + p \cdot 1(y_i \neq \vee_{j=0}^{m} u_{i-j})$$

where '$\vee$' indicates a logical 'OR', and $1(\cdot)$ is the indicator function.
We formed the Ω space as $\{0,1\}^n$, with elements $\omega = (u_0, \cdots, u_{n-1})$. Then each functions $P(y_i|u_i, s_i)$ is measurable in a σ–field $\mathcal{F}_i$, with two atoms $\{\omega : \vee_{j=0}^{m} u_{i-j} = 0\}$ and $\{\omega : \vee_{j=0}^{m} u_{i-j} = 1\}$. Since we like to calculate the posterior probabilities on each input u_i, we also include σ–fields $\mathcal{G}_i$ each with two atoms $\{\omega : u_i = 0\}$ and $\{\omega : u_i = 1\}$.

We then run our algorithm to create a junction tree on the $\mathcal{F}_i$'s and the $\mathcal{G}_i$'s, lifting the space whenever needed. The result is a chain consisting of the $\mathcal{F}_i'$'s with each $\mathcal{G}_i'$ hanging from its corresponding $\mathcal{F}_i'$ (see Figure 6).

We have run the algorithm on different values of n and m. Table 1 compares the complexity of the message-passing algorithm on the probabilistic junction tree and the normal GDL (BCJR) algorithm. To count for the final marginalization at the atoms of the original σ-fields, we have used $5nz$ as the (approximate) arithmetic complexity of our algorithm, where $nz = \sum_{(i,j) \text{ an edge}} nz(i,j)$ is the total number of nonzero elements in the tables of pairwise joint measure along the edges of the tree (see Section 4.3).

The details of the case $n = 12, m = 6$ have been portrayed in Figure 6. The numbers underneath each $\mathcal{F}_i'$ shows the number of atoms of $\mathcal{F}_i' \vee \mathcal{G}_i'$ after lifting has been done. Note that with our setup, originally $\mathcal{F}_0 \vee \mathcal{G}_0$ has 2 atoms, and all other $\mathcal{F}_i \vee \mathcal{G}_i$'s have 3 atoms. The numbers under the brackets denote $nz(i,j)$, the number of nonzero elements in the matrix of joint measures between the atoms of adjacent nodes; As mentioned before, the number of operations required in the computation of the message from one node to an adjacent node is proportional to this number.

(n, m)	(9, 5)	(9, 6)	(9, 7)	(10, 5)	(10, 6)	(10, 7)	(11, 5)	(11, 6)	(11, 7)	(12, 5)	(12, 6)	(12, 7)
nz	95	130	123	107	133	186	119	147	217	129	159	230
GDL ops	1440	2880	5760	1600	3200	6400	1760	3520	7040	1920	3840	7680
PGDL ops	475	650	615	535	665	930	595	735	1085	645	795	1150

Table 1: Comparison between complexity of GDL and probabilistic GDL. Here nz denotes the total number of nonzero entries in the tables of pairwise joint measures. For a chain of length n and memory m, GDL algorithm requires about $5\,n\,2^m$ arithmetic operations, while PGDL requires about $5\,nz$ operations.

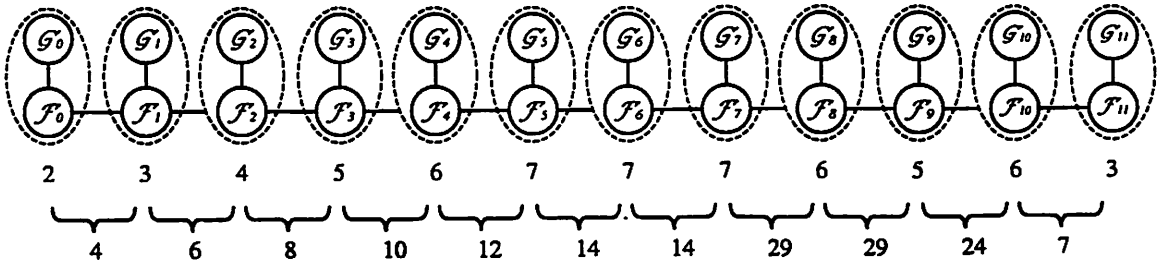

Figure 6: Junction Tree Created for Chain of Length 12 and Memory 6

Example 6. CH-ASIA. Our last example is CH-ASIA from [5], pp. 110-111. The chain graph of Figure 7 describes the dependencies between the variables. Thus the problem is to marginalize $P(S, A, L, T, B, E, D, C, X)$, which is the

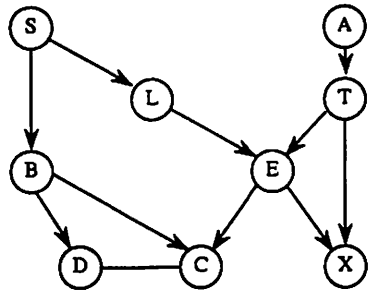

Figure 7: Graph of CH-ASIA Example

product of the following functions: $P(S)$,$P(A)$,$P(L|S)$,$P(T|A)$,$P(B|S)$,$1(E = L \vee T)$,$P(X|E)$,$f(C, D, B)$,$g(C, B, E)$ and $h(B, E)$.

Again we set up measurable spaces, with σ–fields corresponding to each of the above functions. We then ran the lifting algorithm to find a junction tree in form of a chain, as in the previous example. This time, however, due to lack of structure at the level of the marginalizable functions, (i.e. the aforementioned

conditional probabilities,) the algorithm produced exactly a junction tree that one could obtain by the process of moralization and triangulation at the level of original variables. In other words, all liftings were done by addition of one or more 'whole' orthogonal directions (i.e. GDL variables) of the Ω space to the σ–fields. After reconverting σ–fields to 'variables', the junction tree we obtained was the following:

Figure 8: Junction Tree for CH-ASIA Example

In this case, our algorithm has reduced to GDL.

7 Discussion

In this paper we have developed a measure-theoretic version of the junction tree algorithm. We have generalized the notions of independence and junction tree at the level of σ–fields, and have produced algorithms to find or construct a junction tree on a given set of σ–fields. By taking advantage of structures at the atomic level of sample space Ω, our marginalization algorithm is capable of producing solutions far less complex than GDL-type algorithms. The cost of generating a junction tree, however, is exponential in the size of the problem, as is the size of any complete representation of the sample space Ω. Once a junction tree has been constructed, however, the algorithm will only depend on the joint measure of the atoms of adjacent pairs of σ–fields on the tree. This means that an 'algorithm' which was build by considering an Ω space with myriads of elements, can be stored compactly and efficiently.

Using our framework, the tradeoff between the construction complexity of junction trees and the overall complexity of the marginalization algorithm can be made with an appropriate choice for the representation of the measurable spaces; at one extreme, one considers the complete sample space, taking advantage of all the possible structures, and at the other, one represents the sample space with independent variables (i.e. orthogonal directions), in which case our framework reduces to GDL, both in concept and implementation.

The validity of this theory for the *signed* measures is of enormous convenience; it allows for introduction of atoms of negative weight in order to create independencies. It also greatly simplifies the task of lifting, as now it can be done by taking the singular value decomposition of a matrix. In contrast, the problem of finding a *positive* rank-one decomposition of a positive matrix (which would arise if one confined the problem to the positive measures functions) is a very hard problem (See [4]).

References

[1] S.M. Aji and R.J. McEliece, *"The Generalized Distributive Law,"* IEEE Trans. Inform. Theory **46** (no. 2), March 2000, pp.325–343.

[2] F. Baccelli, G. Cohen, G.J. Olsder and J.P. Quadrat, *Synchronization and Linearity*, New York : Wiley, 1992.

[3] L.R. Bahl, J. Cocke, F. Jelinek and J.Raviv, *"Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate,"* IEEE Trans. Inform. Theory **IT-20**, March 1974, pp.284–287.

[4] J.E. Cohen and U.G. Rothblum, *"Nonnegative Ranks, Decompositions, and Factorization of Nonnegative Matrices,"* Linear Algebra Appl. **190**, 1993, pp.149–168.

[5] R.G. Cowell, A.P. Dawid, S.L. Lauritzen and D.J. Spiegelhalter, *Probabilistic Networks and Expert Systems*, New York: Springer-Verlag, 1999.

[6] D.J.C. MacKay and R.M. Neal, *"Good Codes Based on Very Sparse Matrices,"* Cryptography and Coding, 5th IMA conference, Proceedings, pp.100–111, Berlin: Springer-Verlag, 1995

[7] R.J. McEliece, D.J.C. MacKay and J.F. Cheng, *"Turbo Decoding as an Instance of Pearl's 'Belief Propagation' Algorithm,"* IEEE Journal on Selected Areas in Communications, **16** (no. 2), Feb. 1998, pp.140–152.

[8] P. Pakzad and V. Anantharam, *"Conditional Independence for Signed Measures,"* In preparation.

[9] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, San Mateo, CA: Morgan Kaufmann, 1988.

[10] G.R. Shafer and P.P. Shenoy, *"Probability Propagation,"* Ann. Math. Art. Intel., **2**, 1990, pp.327–352.