


A Complete Diagrammatic Calculus for Conditional Gaussian Mixtures

Mateo Torres-Ruiz  

University College London, UK

Robin Piedeleu 

University College London, UK

Alexandra Silva 

Cornell University, Ithaca, NY, USA

Fabio Zanasi 

University College London, UK

Abstract

We extend the synthetic theories of discrete and Gaussian categorical probability by introducing a diagrammatic calculus for reasoning about hybrid probabilistic models in which continuous random variables, conditioned on discrete ones, follow a multivariate Gaussian distribution. This setting includes important families of distributions such as Gaussian mixtures, where each Gaussian component is selected according to a discrete variable. We develop a string diagrammatic syntax for distributions of this type, give it a compositional semantics, and equip it with a sound and complete equational theory that characterises when two mixtures represent the same distribution.

2012 ACM Subject Classification Theory of computation → Categorical semantics; Theory of computation → Logic

Keywords and phrases String diagrams, Category theory, Mixture models, Probability theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2026.11

Related Version *Full Version:* <https://arxiv.org/abs/2510.04649> [28]

Funding This work was partially supported by ARIA’s Safeguarded AI programme. RP, MTR and AS acknowledge support from ERC grant Autoprobe no. 101002697.

Acknowledgements The authors would like to thank the anonymous reviewers for their helpful suggestions.

1 Introduction

Mixture models, particularly Gaussian mixture models (GMMs), are a well-established tool in probabilistic modelling. These are parametric families of probability distributions used as flexible approximations for complex multimodal distributions that appear in diverse applications, ranging from clustering and density estimation to generative modelling and signal processing [3, 18, 27]. Moreover, any continuous distribution can be arbitrarily well approximated by a finite mixture of Gaussians [8]. Despite their ubiquity and expressiveness, reasoning formally about mixtures, especially when two mixtures with different structures define the same distribution, remains a subtle and largely unexplored problem.

This paper develops a compositional, algebraic theory for probability distributions formed from discrete mixture components and multivariate Gaussians, allowing one to reason about their structure and equivalence. Our motivation stems from the observation that mixture models are not just statistical tools but structured expressions that admit algebraic transformations. For instance, a mixture with nested components can often be flattened or reparametrised without changing its semantics, and different symbolic expressions may describe the same underlying distribution. Yet these equivalences are rarely made explicit.



© Mateo Torres-Ruiz, Robin Piedeleu, Alexandra Silva, and Fabio Zanasi; licensed under Creative Commons License CC-BY 4.0

34th EACSL Annual Conference on Computer Science Logic (CSL 2026).

Editors: Stefano Guerrini and Barbara König; Article No. 11; pp. 11:1–11:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The first step to address this gap is to define a formal language for representing probability distributions generated by both discrete and Gaussian variables. Our approach is grounded in the categorical framework of symmetric monoidal categories (SMCs) [17] and their associated language of *string diagrams* [24]. In recent years, SMCs have been identified as a convenient algebraic setting in which to study how probabilistic systems can be composed both in sequence and in parallel [6]. Within this framework, string diagrams offer an intuitive and rigorous graphical syntax for reasoning and manipulating probabilistic expressions. Notably, structural features such as conditional independence, marginalisation, and factorisation can be expressed and manipulated directly in the diagrammatic language [6, 7, 10], making it particularly intuitive for human use. Unlike conventional probabilistic graphical models (PGMs), where diagrams are informal aids to understanding, our diagrams *are* the syntax – they provide a finer-grained perspective on the internal structure of the distributions they represent and come with a formal, compositional semantics, given through a symmetric monoidal functor into a suitable category of probabilistic maps, assigning to each diagram a well-defined stochastic kernel that captures its intended probabilistic meaning. As a result, we can reason not only about the overall distribution but also about how probabilistic dependencies compose and combine in a modular and compositional manner.

Hybrid models that incorporate discrete and continuous random variables have been extensively studied in the context of PGMs. A paradigmatic example is the family of *Conditional Gaussian* distributions (or, simply, CG-distributions), a family of joint distributions where the conditional distribution of continuous variables *given* the discrete ones follows a (multivariate) Gaussian distribution [14, 15]. CG-distributions were first studied in the context of generalised graphical linear models, associated to *conditional linear Gaussian* networks [11] – Bayesian networks that combine discrete and continuous nodes, but in which discrete nodes cannot have continuous parents. Note also that GMMs are a specific case of conditional linear Gaussian networks, in which all of discrete variables are latent.

The main result of this work is an algebraic axiomatisation of diagrammatic equivalence for Gaussian mixtures: a *sound and complete* set of equational rules that captures precisely when two diagrams represent the same distribution. That is, whenever two diagrams encode the same distribution, there is an equational derivation that transforms one into the other. In short, our contribution can be understood as providing a formal syntax, semantics, and complete equational theory for reasoning about CG-distributions in a modular syntax that refines that of conditional linear Gaussian networks. The benefit is two-fold. First, it offers a formal language for verifying that two mixtures define the same distribution without having to compute the semantics explicitly, paving the way to implementation and automation. Second, it supports compositional reasoning: by breaking complex mixture distributions into parts, we can apply *local* equational rules, much like in standard algebra.

A key technical challenge in developing a diagrammatic framework for hybrid models is how to combine two fundamentally different kinds of structure: one that captures discrete (specifically, Boolean) probabilistic variables, and another tailored towards continuous Gaussian variables with conditional dependencies given by affine transformations. Addressing this challenges required us to 1) design a formal syntax in which the two types of variables could interact meaningfully; 2) develop an appropriate *compositional* semantics capable of accommodating both discrete and continuous variables; and 3) identify a sound and complete set of algebraic rules that characterises all the non-trivial interactions between the two fragments. Our solution draws on and generalises two earlier diagrammatic axiomatisations, one for discrete [21], and another for Gaussian kernels [25], integrating the two into a coherent setting for reasoning about hybrid systems.

Outline. In Section 2, we recall the notation, terminology, and mathematical background from (categorical) probability theory, and introduce the semantic domain of interest. Section 3 gives the formal syntax of the diagrams used throughout, motivates its use for both discrete and continuous variables, and Section 4 equips it with a formal semantics capturing their behaviour. In Section 5, we present an equational theory sound for the chosen semantics. Section 6 shows the completeness of this theory via a high-level normalisation argument. Finally, Section 7 concludes with a discussion of future work.

2 Background on (Categorical) Probability Theory

We begin by reviewing the terminology and background from probability theory relevant to our development. As our main objects of study are *mixtures of Gaussians*, which combine discrete and continuous components, we draw on some basic concepts of measure theory. We also cover some of the key ideas of categorical probability theory, building on basic notions of category theory, including that of a SMC. Following this review of standard material, we present our first original contribution: the definition of a (symmetric monoidal) category in which discrete and Gaussian distributions coexist and can be composed in a coherent, unified framework. A passing familiarity with measurable spaces and functions is assumed.

2.1 Stochastic kernels

In what follows, we define the category **Stoch** of measurable spaces and stochastic kernels, following the presentation of Fritz [6, Section 4]. Given two measurable spaces (X, Σ_X) and (Y, Σ_Y) , a *stochastic kernel* $f: (X, \Sigma_X) \rightarrow (Y, \Sigma_Y)$ is a map

$$f: \Sigma_Y \times X \rightarrow [0, 1], \quad (U, x) \mapsto f(U|x)$$

such that $f(\cdot|x): \Sigma_Y \rightarrow [0, 1]$ is a probability distribution for every $x \in X$ and $f(U|\cdot): X \rightarrow [0, 1]$ is a measurable function for every $U \in \Sigma_Y$. The notation $f(U|x)$ indicates that we can think of f as a probability distribution over Y , conditional on X . If $g: (Y, \Sigma_Y) \rightarrow (Z, \Sigma_Z)$ is another stochastic kernel, $g \circ f$ is given by the Chapman-Kolmogorov equation:

$$(g \circ f)(V|x) = \int_{y \in Y} g(V|y) f(dy|x)$$

This defines another stochastic kernel, making measurable spaces and stochastic kernels into a category, **Stoch**, with identity morphisms given by Dirac deltas:

$$\text{id}_X(U|x) = \delta(U|x) = \begin{cases} 1 & \text{if } x \in U, \\ 0 & \text{otherwise.} \end{cases}$$

We call the full subcategory of **Stoch** spanned by finite sets **FinStoch**. Notice that when X and Y are finite sets, any stochastic kernel $f: (X, \mathcal{P}(X)) \rightarrow (Y, \mathcal{P}(Y))$ can be thought of as a (column-)stochastic matrix, *i.e.*, a matrix whose columns sum to one; the composition $g \circ f$ of two stochastic kernels between finite sets then boils down to standard matrix multiplication.

Given two measurable spaces (X_1, Σ_1) and (X_2, Σ_2) , we can form the product space $(X_1 \times X_2, \Sigma_1 \otimes \Sigma_2)$, where $\Sigma_1 \otimes \Sigma_2$ is the σ -algebra generated by the rectangle subsets, *i.e.*, those of the form $U_1 \times U_2$ for $U_1 \in \Sigma_1$ and $U_2 \in \Sigma_2$. Moreover, given two measures μ_1 and μ_2 over (X_1, Σ_1) and (X_2, Σ_2) respectively, we can form the product measure $\mu_1 \otimes \mu_2$ uniquely defined by $(\mu_1 \otimes \mu_2)(U_1 \times U_2) = \mu_1(U_1)\mu_2(U_2)$. This construction extends to stochastic kernels, whose product we characterise as follows on rectangle subsets:

$$f_1 \otimes f_2: (X_1 \times X_2, \Sigma_1 \otimes \Sigma_2) \rightarrow (Y_1 \times Y_2, \Omega_1 \otimes \Omega_2), \quad (U_1 \times U_2 \mid x_1, x_2) \mapsto f_1(U_1|x_1)f_2(U_2|x_2)$$

The unit for this monoidal structure is the singleton set $1 = \{\bullet\}$ equipped with its powerset as σ -algebra. This equips **Stoch** with the structure of a *symmetric monoidal category* $(\mathbf{Stoch}, \otimes, 1, \sigma)$, with symmetry $\sigma_{X,Y}: (X \times Y, \Sigma_X \otimes \Sigma_Y) \rightarrow (Y \times X, \Sigma_Y \otimes \Sigma_X)$ defined by $\sigma(V \times U|x, y) = \delta_x(U) \otimes \delta_y(V)$. **FinStoch** is then a symmetric monoidal subcategory of **Stoch**.

Beyond finite sets, in this work, we will only consider one other class of measurable spaces: the space \mathbb{R}^n equipped with its Borel σ -algebra $\mathcal{B}(\mathbb{R}^n)$. Note that $\mathcal{B}(\mathbb{R}^m) \otimes \mathcal{B}(\mathbb{R}^n) \cong \mathcal{B}(\mathbb{R}^{m+n})$. We will use this isomorphism implicitly whenever it is convenient.

2.2 Gaussian Probability

We review here the basics of Gaussian distributions and their transformation under affine maps [6, 13]. For a more detailed account, we refer the reader to the extensive literature on the subject [2, 23]. Gaussian distributions – aka *normal distributions* – are one of the most fundamental classes of continuous probability distributions for real-valued random variables.

Univariate Gaussians. A random variable \mathbf{X} is *Gaussian* with mean $\mu \in \mathbb{R}$ and variance σ^2 with $\sigma \in [0, \infty)$ if it has a probability density function given by $f(x | \mu, \sigma^2) = \{2\pi\sigma^2\}^{-1/2} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$ for $x \in \mathbb{R}$ with respect to the Lebesgue measure. This is typically denoted by $\mathbf{X} \sim \mathcal{N}(\mu, \sigma^2)$.

Multivariate Gaussians. Normal distributions generalise naturally to high dimensional spaces by pushing forward products of independent *standard normal distributions*, $\mathcal{N}(0, 1)$ under some affine transformation.

► **Definition 1** (Multivariate Gaussian distribution). *A k -dimensional random vector \mathbf{X} follows a multivariate Gaussian distribution (k -variate) if every linear combination of its components is normally distributed. Equivalently, \mathbf{X} can be expressed as $\mathbf{X} = A \cdot \mathbf{Z} + \mu$, where $A \in \mathbb{R}^{k \times n}$ has rank n , $\mu \in \mathbb{R}^k$ and the random vector \mathbf{Z} has components $Z_1, \dots, Z_n \sim \mathcal{N}(0, 1)$. We call μ the mean vector and $\Sigma := AA^T$ the covariance matrix of \mathbf{X} , and write $\mathbf{X} \sim \mathcal{N}_k(\mu, \Sigma)$.*

Similar to the univariate case, a multivariate Gaussian $\mathbf{X} \sim \mathcal{N}_k(\mu, \Sigma)$ is fully characterised by its mean vector $\mu \in \mathbb{R}^k$ and its covariance matrix, $\Sigma \in \mathbb{R}^{k \times k}$, representing the expectation and the pairwise covariances of the variables. Note that Σ is necessarily *positive semi-definite* and can be factored as $\Sigma = LL^T$ for some lower triangular matrix L (a Cholesky decomposition of Σ). We also write $\mathcal{N}_k(\mu, \Sigma)(Y)$ for the probability of an event $Y \in \mathcal{B}(\mathbb{R}^k)$. Note that, when the covariance matrix is 0, the corresponding measure is a Dirac at its mean, $\delta(\cdot | \mu)$.

Gaussian maps. We are interested in particularly simple stochastic kernels comprising an affine transformation with additive Gaussian noise. Formally, a Gaussian map is a stochastic kernel $f: \mathbb{R}^m \rightarrow \mathbb{R}^n$ for which there exists an $n \times m$ matrix A , a vector $b \in \mathbb{R}^n$ and a positive semi-definite $n \times n$ matrix Σ such that $f(\cdot|x) \sim \mathcal{N}_n(Ax + b, \Sigma)$ ¹; we write $f(x) = \mathcal{N}_n(Ax + b, \Sigma)$. The composite of two Gaussian maps in **Stoch** is still a Gaussian map: with f as before, and $g: \mathbb{R}^n \rightarrow \mathbb{R}^k$ given by $g(y) = \mathcal{N}_k(Cy + d, \Theta)$, then $(g \circ f)(x) = \mathcal{N}_k(CAx + Cb + d, C\Sigma C^T + \Theta)$. Thus, Gaussian maps form a subcategory of

¹ Here, we see $f(\cdot|x)$ as a random variable with value in \mathbb{R}^n .

Stoch [6]. In fact, they form a symmetric monoidal subcategory of **Stoch** with the monoidal product given as in Section 2.1: for two Gaussian maps $f_1(x) = \mathcal{N}_{n_1}(A_1x + b_1, \Sigma_1)$ and $f_2(x) = \mathcal{N}_{n_2}(A_2x + b_2, \Sigma_2)$, then for $x_1 \in \mathbb{R}^{m_1}$ and $x_2 \in \mathbb{R}^{m_2}$, let

$$(f_1 \otimes f_2) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \mathcal{N}_{n_1+n_2} \left(\begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix} \right)$$

2.3 Mixtures of Gaussians

In this work, we are concerned with mixtures: convex combinations of several component distributions. Intuitively, sampling from a mixture proceeds by first choosing a component based on the mixture weights, then sampling from the chosen component.

► **Definition 2** (Mixture distribution). *A mixture distribution is a finite convex combination of distributions $\{\mu_i\}_{i=1,\dots,k}$ over the same measurable space (X, Ω) , i.e., a measure μ defined by $\mu(E) = \sum_{i=1}^k p_i \cdot \mu_i(E)$, where $\sum_{i=1}^k p_i = 1$ and $p_i \geq 0$, for a measurable set $E \in \Omega$.*

In what follows we will be concerned with measurable spaces $(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n))$ for some natural number n , equipped with their usual Borel σ -algebra $\mathcal{B}(\mathbb{R}^n)$ where the component distributions μ_i are all (multivariate) Gaussians – we call these *mixtures of Gaussians*. Note that a mixture of Gaussians is different from taking the convex sum of several Gaussian random variables, which is still Gaussian. In what follows, we will make extensive use of the following unique characterisation result.

► **Proposition 3.** *Mixtures of Gaussians $\sum_i p_i \cdot \mathcal{N}_k(\mu_i, \Sigma_i)$ are uniquely determined by their parameters, i.e., the mixture weights p_i , component means μ_i , and covariances Σ_i .*

2.4 Conditional Gaussian mixtures

The first original contribution of our paper is the definition of a SMC that unifies **FinStoch**, the category of stochastic kernels between finite sets (specifically, powers of the two-element set, \mathbb{B}^n , for some natural number n , with $\mathbb{B} := \{0, 1\}$), and **Gauss**, the category of Gaussian maps. Morphisms of this category will be stochastic kernels of type $\mathbb{B}^p \otimes \mathbb{R}^m \rightarrow \mathbb{B}^q \otimes \mathbb{R}^n$ satisfying certain conditions. Given such a kernel $f: \mathbb{B}^p \otimes \mathbb{R}^m \rightarrow \mathbb{B}^q \otimes \mathbb{R}^n$, $a \in \mathbb{B}^p$, $x \in \mathbb{R}^m$, and $b \in \mathbb{B}^q$, let $f(\cdot|b, a, x)$ be the distribution over \mathbb{R}^n obtained by conditioning f on the discrete output component being equal to b . The morphisms of our category are those for which the conditional $f(\cdot|b, a, x)$ is a mixture of Gaussians (if the probability of obtaining b is zero, the conditional may be defined arbitrarily as a mixture of Gaussians). This implies that we can find some finite set I and stochastic kernels $\varphi: \mathbb{B}^p \rightarrow I \otimes \mathbb{B}^q$, $f_i: \mathbb{B}^p \otimes \mathbb{R}^m \rightarrow \mathbb{R}^n$, such that $f_i(a, x) = A_i(a)x + \mathcal{N}_n(\mu_i(a), \Sigma_i(a))$, for all $i \in I$, $a \in \mathbb{B}^p$, $x \in \mathbb{R}^m$ and, for all $B \subseteq \mathbb{B}^q$, $Y \in \mathcal{B}(\mathbb{R}^n)$, we have

$$\begin{aligned} f(B \times Y | a, x) &= \sum_{b \in B} \sum_{i \in I} \varphi(i, b|a) \cdot f_i(Y|a, x) \\ &= \sum_{b \in B} \sum_{i \in I} \varphi(i, b|a) \cdot \mathcal{N}_n(A_i(a)x + \mu_i(a), \Sigma_i(a))(Y) \end{aligned} \tag{1}$$

► **Definition 4** (Conditional Gaussian mixture). *We call conditional Gaussian mixture (CG-mixture) a stochastic kernel $f: \mathbb{B}^p \otimes \mathbb{R}^m \rightarrow \mathbb{B}^q \otimes \mathbb{R}^n$ expressible as in (1).*

The core idea is that CG-mixtures represent joint distributions over discrete and continuous variables, such that the conditional distribution over the continuous variables – given fixed values of the discrete variables in its domain – is a Gaussian mixture. The following is a consequence of Proposition 3.

► **Proposition 5.** *CG-mixtures are uniquely determined by their parameters, i.e., in the notation above, the set I and the stochastic kernels $\varphi, f_i(a, x)$.*

In light of the previous proposition, we use the notation $f(a, x) = \sum_{i \in I} \varphi(i, \cdot | a) \cdot \mathcal{N}_n(A_i(a)x + \mu_i(a), \Sigma_i(a))$ for CG-mixtures. Note that any Gaussian map can be seen as a CG-mixture (with a single component); similarly, any stochastic kernel between finite sets can be seen as a CG-mixture, with trivial Gaussian components $\mathcal{N}_0(\bullet, ())$, the only distribution over $\mathbb{R}^0 \cong 1 = \{\bullet\}$ with covariance the unique 0×0 matrix. Finally, any mixture of Gaussians $\sum_i p_i \cdot \mathcal{N}_n(\mu_i, \Sigma_i)$ is a CG-mixture $f: 0 \rightarrow \mathbb{R}^n$ given by $f(\bullet, \bullet) = \sum_i \varphi(i, \cdot | \bullet) \cdot \mathcal{N}_n(\mu_i, \Sigma_i)$ where $\varphi(i, \cdot | \bullet) = p_i$.

A related definition has appeared in the work of Lauritzen, in which Bayesian networks with discrete and Gaussian nodes are considered [13]. He calls *Conditional Gaussian* a distribution over sets of discrete and continuous random variables, such that the conditional distribution of the continuous variables given the discrete ones is Gaussian. Our approach recasts these ideas categorically. Importantly, we work in a variable-free setting, where latent variables can be thought of as the index set of the mixture and the observable discrete variables as its inputs/outputs. This is also why our main semantic object of interest, CG-mixtures, are mixtures and not simply Gaussians: as we saw, composing stochastic maps involves marginalising over the intermediate variables (which become latent), so that composing conditional Gaussians in the sense of Lauritzen returns a mixture in general (and not necessarily a single Gaussian).

► **Proposition 6.** *CG-mixtures are closed under composition and monoidal product in Stoch .*

As a result, CG-mixtures form a symmetric monoidal subcategory of Stoch (symmetries and identities of the relevant type can readily be seen to be CG-mixtures).

► **Definition 7 (MixGauss).** *We call MixGauss the symmetric monoidal subcategory of $(\text{Stoch}, \otimes, \sigma)$ whose objects are measurable spaces of the form $\mathbb{B}^q \otimes \mathbb{R}^n$ for some $q, n \in \mathbb{N}$ and whose morphisms are CG-mixtures.²*

3 String diagrammatic syntax

Our main goal is to provide a sound and complete equational theory for CG-mixtures, which faithfully capture both discrete (Boolean) and continuous (Gaussian) variables, as well as their interaction. We approach this problem by first giving a modular syntax for these distributions in terms of string diagrams, a standard graphical language used to represent morphisms in SMCs. While we review some of the basics of string diagrams below, we refer the reader to [22] for a more comprehensive introduction.

The two-dimensional syntax we use is a formal, graphical notation for morphisms of a *coloured product and permutation category* (or simply, a *prop*). A prop is a strict SMC whose objects are the set C^* of words over a (finite) number of *colours* C and the monoidal product is given by concatenation.

The prop that will serve as our syntax P_Σ is freely generated from a *monoidal signature* Σ , a set of generating morphisms $g: v \rightarrow w$, through *sequential* ($c; d$) and *parallel* ($c \otimes d$) composition of generators of appropriate type, together with identities $\text{—} : c \rightarrow c$ for $c \in C$,

² The reader may notice that $(\mathbb{B}^p \otimes \mathbb{R}^m) \otimes (\mathbb{B}^q \otimes \mathbb{R}^n) \neq \mathbb{B}^{p+q} \otimes \mathbb{R}^{m+n}$, but that the two sides are merely isomorphic. In what follows, we will behave as if they were equal, assuming implicitly that we can use the isomorphisms $\mathbb{B}^p \otimes \mathbb{B}^q \cong \mathbb{B}^{p+q}$ and $\mathbb{R}^m \otimes \mathbb{R}^n \cong \mathbb{R}^{m+n}$ wherever necessary. This defines a *strict* SMC, which is what we need as semantics to our diagrammatic syntax, also a strict SMC.

symmetry $\bowtie : cd \rightarrow dc$ for $c, d \in C$ and empty generator $\square : \varepsilon \rightarrow \varepsilon$ (where ε denotes the empty word). The two binary operations used to construct new terms, $(-; -) : P_\Sigma \times P_\Sigma \rightarrow P_\Sigma$ and $(- \otimes -) : P_\Sigma \times P_\Sigma \rightarrow P_\Sigma$ are the categorical composition and the monoidal product of the prop, respectively. The morphisms of P_Σ are thus terms of a $C^* \times C^*$ -sorted syntax quotiented by the laws of SMCs:

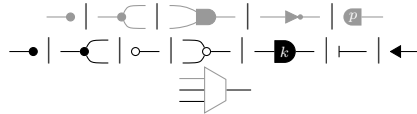
$$\begin{aligned} c_1 \otimes (c_2 \otimes c_3) &= (c_1 \otimes c_2) \otimes c_3 & (c_1 \otimes c_2); (d_1 \otimes d_2) &= (c_1; d_1) \otimes (c_2; d_2) \\ (c; d); e = c; (d; e) & & c; \text{---} = c = \text{---}; c & & \square \otimes c = c = c \otimes \square \\ (\text{---} \otimes c); \bowtie = \bowtie; (c \otimes \text{---}) & & \bowtie; \bowtie = \text{---} \otimes \text{---} & & \end{aligned}$$

where c, d, e and c_i, d_i range over Σ -terms of the appropriate type (omitted for clarity). These axioms state that the two forms of composition are associative and unital, that they satisfy a form of interchange law, and that the wire crossings behave as expected. Each Σ -term c of type $v \rightarrow w$ will be graphically represented as a string diagram with labelled wires $\begin{matrix} v \\ \boxed{c} \\ w \end{matrix}$. For $c : u \rightarrow v, d : v \rightarrow w, c_i : v_i \rightarrow w_i$, we depict $c; d$ as sequential composition and $c_1 \otimes c_2$ as parallel composition of Σ -terms of type $u \rightarrow w$ and $v_1 v_2 \rightarrow w_1 w_2$,

$$\begin{matrix} u \\ \boxed{c; d} \\ w \end{matrix} = \begin{matrix} u \\ \boxed{c} \\ v \end{matrix} \begin{matrix} v \\ \boxed{d} \\ w \end{matrix} \quad \begin{matrix} v_1 v_2 \\ \boxed{c_1 \otimes c_2} \\ w_1 w_2 \end{matrix} = \begin{matrix} v_1 \\ \boxed{c_1} \\ w_1 \end{matrix} \begin{matrix} v_2 \\ \boxed{c_2} \\ w_2 \end{matrix}$$

The signature Σ over which we generate our graphical syntax contains

- two colours, B and R, whose identities we depict respectively as --- and --- respectively;
- the following generating morphisms:



These can be seen as the constants of our language, and indeed will be the basic components of the diagrams that make up our syntax. Intuitively, our string diagrams will be freely formed much like conventional circuits, by wiring the above generators in sequence or in parallel, crossing wires (with the symmetries, $\bowtie, \bowtie, \bowtie, \bowtie$, etc.) and making them as long as required (with identities, --- or ---). We refer to the resulting two-coloured prop as **MixCirc**, and its morphisms as *mixed circuits* or simply *circuits*. We write B^p (resp. R^n) to denote a word containing p (resp. n) successive B (resp. R). A circuit $c : B^p R^m \rightarrow B^q R^n$ will be depicted using a white box $\begin{matrix} p & & q \\ \text{---} & \boxed{c} & \text{---} \\ m & & n \end{matrix}$, in which the first p input and q output wires have Boolean type B, while the remaining m input and n output wires denote the real type R. To simplify our syntax, we will sometimes avoid labelling wires explicitly: in this case, unlabelled thick wires represent an arbitrary number of wires (including potentially zero wires), while a normal-width wire indicates a single wire.

Although we have not yet defined a formal semantics for interpreting these, the colours and shapes hint at their intended meaning. The *grey generators* resemble traditional Boolean circuit gates with the addition of a *probabilistic gate*, --- , which emits a T (true) with probability p and a F (false) with probability $1 - p$. The remaining grey generators correspond to the standard logical gates of conjunction (---) and negation (---), together with a copying gate, --- , that broadcasts its input to two output wires, and a terminating wire, --- , that discards any input.

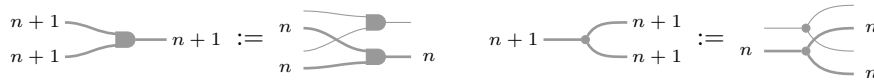
We will refer to string diagrams in this fragment as (*probabilistic*) *Boolean circuits*, and use grey boxes to represent a generic Boolean circuit $B^q \rightarrow B^p$. Note that such circuits have appeared in previous work, equipped with a sound and complete axiomatisation [21].

Similarly, the *black generators* are interpreted as processing real values. The preliminary intuition for the *copier*, --- , and *discarder*, --- , is the same as their Boolean counterpart; the generator --- denotes addition, --- produces the value 0, --- produces the value 1, and

\bullet multiplies by the scalar $k \in \mathbb{R}$. This set of generators has previously been used to give a compositional syntax for *affine maps* [4]. The additional generator \leftarrow is interpreted as sampling values randomly from a *standard normal distribution*, $\mathcal{N}(0, 1)$. Together, these allow us to represent Gaussian maps [25], as we will explain in the following section. We will refer to string diagrams in this fragment as *Gaussian circuits*, denote the corresponding prop as **GaussCirc**, and use black boxes to represent generic circuits in this fragment.

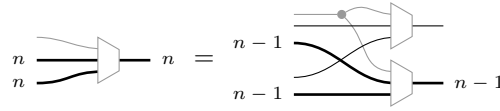
The last generator, \equiv , serves as the interface between the Boolean and Gaussian fragments of our syntax. It behaves like an *if-then-else* gate, which selects the value of its output wire based on the value of the discrete one, which we call the *guard*: if it is true, the gate outputs the second input; if it is false, it outputs its third input.

► Remark 8. We extend the syntactic sugar given by the below thick wires to represent multiple instances of these generators,



and analogously with \rightarrow and \bullet .

Finally, we will also make use of n -ary if-then-else circuits (for $n > 1$), defined as the composite of n if-then-else generators which share a single guard:



4 Semantics

We define the semantics of circuits as a mapping $\llbracket \cdot \rrbracket$ from the generators to stochastic kernels – specifically, to CG-mixtures. Since the diagrammatic syntax is freely-generated, the mapping will extend to a symmetric monoidal functor into **MixGauss** (Definition 7), with which we can compute the behaviour of arbitrary composite circuits.

Note that, for the purely Boolean or Gaussian fragments, our semantics coincides with that given in previous work: Boolean circuits are interpreted as stochastic kernels between sets of the form \mathbb{B}^p [21], while Gaussian circuits are interpreted as Gaussian maps [25].

We write $()$ for the unique 0×0 matrix, \bullet for the unique element of \mathbb{R}^0 and \mathbb{B}^0 , and $\delta(\cdot | x)$ for the Dirac delta at x . Each generator $g: \mathbb{B}^p \mathbb{R}^m \rightarrow \mathbb{B}^q \mathbb{R}^n$ is interpreted as a CG-mixture $\llbracket g \rrbracket: \mathbb{B}^p \otimes \mathbb{R}^m \rightarrow \mathbb{B}^q \otimes \mathbb{R}^n$, specified as a mixture $\llbracket g \rrbracket(a, x) = \sum_{i \in I} \varphi(i, \cdot | a) \cdot \mathcal{N}_n(A_i(a)x + \mu_i(a), \Sigma_i(a))$ introduced in Section 2.4. Notice that none of our generators represents a non-trivial mixture of Gaussians – these will arise by composing them. Since for all the generators, either the Boolean or the real output component is trivial, we omit the irrelevant part of the expression, to avoid notational overload. For example, when $n = 0$ (no real-valued output), we simply write $\llbracket g \rrbracket(a, x) = \sum_i \varphi(i, \cdot | a) = \sum_i p_i \cdot \delta(\cdot | i)$ for some weights $p_i \in [0, 1]$ that add to one. Moreover, when $q = 0$ (no Boolean-valued output) and $I = 1$ (trivial mixture), the resulting distribution is a single Gaussian map, which we write simply as $\llbracket g \rrbracket(a, x) = \mathcal{N}(Ax + \mu, \Sigma)$. With these notational preliminaries out of the way, we are ready to specify the semantics of our generators:

$$\llbracket \equiv \rrbracket \left(b, \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right) := \begin{cases} \mathcal{N}_1(x_1, 0) & \text{if } b = 1 \\ \mathcal{N}_1(x_2, 0) & \text{if } b = 0 \end{cases}$$

$$\begin{array}{ll}
\llbracket \text{---} \bullet \rrbracket (b, \bullet) := \delta(\cdot | \bullet) & \llbracket \text{---} \text{---} \rrbracket (b, \bullet) := \delta\left(\cdot \left| \begin{pmatrix} b \\ b \end{pmatrix} \right.\right) \\
\llbracket \text{---} \text{---} \rrbracket \left(\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \bullet\right) := \delta(\cdot | b_1 \wedge b_2) & \llbracket \text{---} \text{---} \rrbracket (b, \bullet) := \delta(\cdot | \neg b) \\
\llbracket \text{---} \text{---} \rrbracket (\bullet, \bullet) := p \cdot \delta(\cdot | 1) + (1-p) \cdot \delta(\cdot | 0) & \llbracket \text{---} \text{---} \rrbracket (\bullet, \bullet) := \mathcal{N}_1(0, 1) \\
\llbracket \text{---} \bullet \rrbracket (\bullet, x) := \mathcal{N}_0(\bullet, ()) & \llbracket \text{---} \text{---} \rrbracket (\bullet, x) := \mathcal{N}_2\left(\begin{pmatrix} x \\ x \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}\right) \\
\llbracket \text{---} \text{---} \rrbracket (\bullet, \bullet) := \mathcal{N}_1(0, 0) & \llbracket \text{---} \text{---} \rrbracket \left(\bullet, \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) := \mathcal{N}_1(x_1 + x_2, 0) \\
\llbracket \text{---} \text{---} \rrbracket (\bullet, x) := \mathcal{N}_1(k \cdot x, 0) & \llbracket \text{---} \text{---} \rrbracket (\bullet, \bullet) := \mathcal{N}_1(1, 0)
\end{array}$$

■ **Figure 1** Semantics of circuit generators in terms of CG-mixtures.

Recall also that the degenerate Gaussian $\mathcal{N}_n(\mu, 0)$ is equal to $\delta(\cdot | \mu)$, the Dirac at μ . For example, $\llbracket \text{---} \rrbracket (\bullet, \bullet) := \mathcal{N}_1(1, 0)$ could have also been written as $\delta(\cdot | 1)$ (and the same goes for the interpretations of --- , --- , --- , --- , ---). The following is a simple consequence of the freeness of the diagrammatic syntax.

► **Proposition 9.** $\llbracket \cdot \rrbracket$ extends to a symmetric monoidal functor $\text{MixCirc} \rightarrow \text{MixGauss}$.

This means that each circuit $\text{---} \text{---} \text{---}$ can be assigned a morphism $\llbracket \text{---} \text{---} \rrbracket : \mathbb{B}^p \otimes \mathbb{R}^m \rightarrow \mathbb{B}^q \otimes \mathbb{R}^n$ of MixGauss where p (q) is the number of occurrences of \mathbf{B} in v (w) and m (n) is the number of occurrences of \mathbf{R} in v (w). Crucially, the previous proposition states that this assignment can be made *compositional* – in other words, the semantics of any composite circuit can be computed from the semantics of the generators above following the expressions for composition and the monoidal product in MixGauss (which boil down to composition in Stoch , cf. Section 2). In particular, when $\llbracket \text{---} \text{---} \rrbracket (a, x) = \sum_{i \in I} \varphi(i, \cdot | a) \cdot \mathcal{N}_m(A_i(a)x + \mu_i(a), \Sigma_i(a))$ and $\llbracket \text{---} \text{---} \rrbracket (b, y) = \sum_{j \in J} \psi(j, \cdot | b) \cdot \mathcal{N}_n(B_j(b)y + \nu_j(b), \Theta_j(b))$ their composition is given by

$$\llbracket \text{---} \text{---} \rrbracket (a, x) = \sum_{(i,j,b) \in I \times J \times B} \pi(i, j, b, \cdot | a) \cdot h_{i,j,b}(a, x)$$

where $h_{i,j,b}(a, x) := \mathcal{N}_n(B_j(b)A_i(a)x + B_j(b)\mu_i(a) + \nu_j(b), B_j(b)\Sigma_i(a)B_j(b)^T + \Theta_j(b))$ and $\pi(i, j, b, \cdot | x) := \psi(j, \cdot | b)\varphi(i, b | a)$. Intuitively speaking, for every pair of Gaussian maps f_i and g_j of the two mixtures $\llbracket c \rrbracket$ and $\llbracket d \rrbracket$, we compose them *qua* Gaussian maps (as explained in Section 2) and assign the product of the weights of their respective weights to the composite $g_j \circ f_i$ in the resulting mixture. Moreover, when $\llbracket \text{---} \text{---} \rrbracket (a, x) = \sum_{i \in I} \varphi(i, \cdot | a) \cdot \mathcal{N}_m(A_i(a)x + \mu_i(a), \Sigma_i(a))$ and $\llbracket \text{---} \text{---} \rrbracket (b, y) = \sum_{j \in J} \psi(j, \cdot | b) \cdot \mathcal{N}_o(B_j(b)y + \nu_j(b), \Theta_j(b))$

their monoidal product $\llbracket \text{---} \text{---} \rrbracket \left(\begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} x \\ y \end{pmatrix}\right)$ is given by

$$\sum_{(i,j) \in I \times J} \varphi(i, \cdot | a)\psi(j, \cdot | b) \cdot \mathcal{N}_{m+o}\left(\begin{pmatrix} A_i(a) & 0 \\ 0 & B_j(b) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \mu_i(a) \\ \nu_j(b) \end{pmatrix}, \begin{pmatrix} \Sigma_i(a) & 0 \\ 0 & \Theta_j(b) \end{pmatrix}\right)$$

11:10 A Complete Diagrammatic Calculus for Conditional Gaussian Mixtures

► **Example 10.** We illustrate how to form mixtures of univariate Gaussians using this syntax. First, the following circuit allows us to take convex sum of its (real-valued) inputs:

$$\left[\left[\begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \right] \right] \left(\begin{array}{c} x_1 \\ x_2 \end{array} \right) = p \cdot \mathcal{N}_1(x_1, 0) + (1-p) \cdot \mathcal{N}_1(x_2, 0) = p \cdot \delta(\cdot | x_1) + (1-p) \cdot \delta(\cdot | x_2)$$

To form convex mixtures of Gaussians, we first need to understand how Gaussian circuits allow us to represent (multivariate) Gaussians, following [25]. As we saw, \leftarrow represents a single univariate centred and normalised Gaussian variable, $\mathcal{N}_1(0, 1)$. By multiplying it by some $\sigma \in \mathbb{R}$, we scale the variance by σ^2 , obtaining a circuit $\leftarrow \sigma$ encoding $\mathcal{N}_1(0, \sigma^2)$; if we want to shift the mean by some value $\mu \in \mathbb{R}$, we can use \curvearrowright to add μ . For example,

$$\left[\left[\begin{array}{c} \leftarrow \mu \\ \leftarrow \sigma \end{array} \right] \right] = \mathcal{N}_1(\mu, \sigma^2) \quad \left[\left[\begin{array}{c} \leftarrow \mu \\ \leftarrow \sigma \end{array} \right] \right] = \mathcal{N}_1(0, \sigma^2)$$

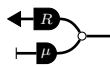
We can now take the p -mixture of the two circuits above:

$$\left[\left[\begin{array}{c} \leftarrow \mu \\ \leftarrow \sigma \\ \leftarrow \mu \\ \leftarrow \sigma \end{array} \right] \right] = p \cdot \mathcal{N}_1(\mu, \sigma^2) + (1-p) \cdot \mathcal{N}_1(0, \sigma^2)$$

For *multivariate* Gaussians, we first need to recall how matrices (and vectors, which are just a special case) are encoded in graphical linear algebra [4]. An $n \times m$ matrix A may be represented by a circuit $\overset{m}{\leftarrow} A \overset{n}{\rightarrow}$, using \leftarrow , $\leftarrow x$, \curvearrowright as follows: its input wires stand for the columns of A , its output wires stand for the rows, and the j -th input wire is connected to the i -th output wire through a scalar $\leftarrow x$ if the coefficient A_{ij} is x . Moreover, since $\leftarrow 0$ is semantically equal to \bullet and $\leftarrow 1$ to --- , we can depict $A_{ij} = 0$ by not connecting the two input/output wires, and $A_{ij} = 1$ by a plain wire.

$$\text{For instance, for } A = \begin{pmatrix} x_1 & 0 & 0 \\ 1 & 1 & 0 \\ x_2 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ let } \overset{m}{\leftarrow} A \overset{n}{\rightarrow} := \begin{array}{c} \text{---} \\ \leftarrow x_1 \\ \leftarrow x_2 \\ \text{---} \end{array}$$

Then, $\left[\left[\overset{m}{\leftarrow} A \overset{n}{\rightarrow} \right] \right] (x) = \mathcal{N}_n(Ax, 0) = \delta(\cdot | Ax)$, justifying the encoding. Now we can encode any multivariate Gaussian $\mathcal{N}_n(\mu, \Sigma)$ as the circuit



with $\Sigma = RR^T$, *i.e.* some Cholesky decomposition of the covariance matrix. Then,

$$\left[\left[\begin{array}{c} \leftarrow R_1 \\ \leftarrow \mu_1 \\ \leftarrow R_2 \\ \leftarrow \mu_2 \end{array} \right] \right] = p \cdot \mathcal{N}_n(\mu_1, R_1 R_1^T) + (1-p) \cdot \mathcal{N}_n(\mu_2, R_2 R_2^T)$$

Moreover, this can be iterated to mixtures with more components in a straightforward way. For example, for three components, we have

$$\left[\left[\begin{array}{c} \leftarrow R_1 \\ \leftarrow \mu_1 \\ \leftarrow R_2 \\ \leftarrow \mu_2 \\ \leftarrow R_3 \\ \leftarrow \mu_3 \end{array} \right] \right] = p_1 \cdot \mathcal{N}_n(\mu_1, R_1 R_1^T) + p_2 \cdot \mathcal{N}_n(\mu_2, R_2 R_2^T) + p_3 \cdot \mathcal{N}_n(\mu_3, R_3 R_3^T)$$

where $p = p_1$ and $q = \frac{p_2}{1-p_1}$. We will make use of this representation when defining a normal form for circuits below.

From this example, it is easy to see that *any* mixture of Gaussians can be encoded as a circuit; in fact, this is true of any CG-mixture: for $\llbracket f \rrbracket(a, x) = \sum_{i \in I} \varphi(i, \cdot | a) \cdot \mathcal{N}_n(A_i(a)x + \mu_i(a), \Sigma_i(a))$, we can encode φ as a Boolean circuit and the individual Gaussian maps as Gaussian circuits [25]. Then, we can put these components together using if-then-else gates arranged in sequence as above, in order to select the right component for each Boolean input and each index $i \in I$.

► **Proposition 11.** *For any CG-mixture f , there exists a circuit c such that $\llbracket c \rrbracket = f$.*

5 Equational Theory

While the laws of SMCs capture some of the semantic equivalences between circuits, they are not sufficient to derive all valid equivalences. To address this, we introduce a additional set of equations, ruling the interactions between the different components of our syntax. For more background on equational theories of string diagrams, aka symmetric monoidal theories, we refer the reader to a more detailed introductory text [22]. For a given theory \mathbb{T} , we write $\stackrel{\mathbb{T}}{=}$ for an equality which can be derived by applying a finite sequence of axioms of \mathbb{T} . Following standard terminology, we say that \mathbb{T} is *sound* (for a given semantics $\llbracket \cdot \rrbracket$) when, for any two circuits c, d of the same type, $c \stackrel{\mathbb{T}}{=} d$ implies that $\llbracket c \rrbracket = \llbracket d \rrbracket$. In other words, two circuits can be shown equal only if they denote the same stochastic kernel. When the converse is true – $\llbracket c \rrbracket = \llbracket d \rrbracket$ implies $c \stackrel{\mathbb{T}}{=} d$ – we say that the theory is *complete*. A sound and complete theory is said to *axiomatise* the corresponding domain of interpretation.

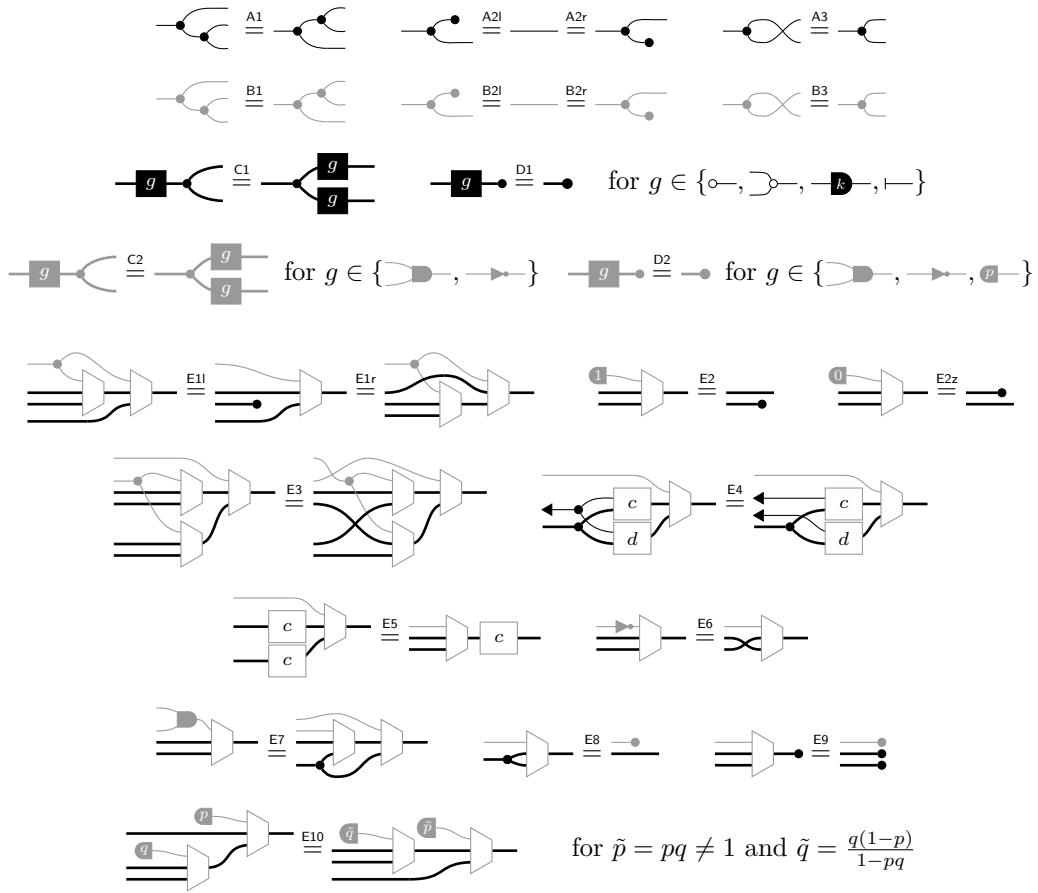
In this work, we are interested in the equational theory that rules the interaction of the Boolean and Gaussian fragments of our syntax, since each of these two individual fragments have already been axiomatised in previous work: probabilistic Boolean circuits [21] and Gaussian circuits [25]. For this reason, we will always assume that two probabilistic Boolean (resp. Gaussian) circuits can be shown to be equal when they have the same interpretation, that is, when they denote the same stochastic kernel (appealing implicitly to the completeness results of the two cited papers). We will also assume that any stochastic kernel of type $\mathbb{B}^p \rightarrow \mathbb{B}^q$ and any Gaussian map $\mathbb{R}^m \rightarrow \mathbb{R}^n$ can be encoded as a circuit in the corresponding fragment – again, this follows from the cited results [21, 25].

The main result of this paper is the soundness and completeness of the theory given in Fig. 2 (relative to the completeness of the Boolean and Gaussian fragments). In what follows, we call this theory **CGM**. We will justify its soundness in this section (Theorem 12); its completeness is more challenging and will be the object of the next section. A few comments are in order to clarify the meaning of the axioms of **CGM**. Axioms **A1-A3** encode the co-associativity, co-unitality, and co-commutativity of the copying ($\dashv\!\!\dashv\!\!\dashv$) and discarding ($\dashv\!\!\dashv\!\!\dashv$) generators for real values. These are common structure in many diagrammatic languages, guaranteeing that the different ways of sharing some value between different parts of a diagram are equal to one another. Immediately below, we have similar axioms **B1-B3** for the same operations on Boolean values.

The next two lines encode the interaction of the other generators with copying and discarding. Axioms **C1, C2** allow us to copy only those generators which denote *deterministic* maps, *i.e.*, those stochastic maps that, given their inputs, return a Dirac delta at their output. For example, $g = \dashv\!\!\dashv\!\!\dashv$ represents addition, a deterministic operation that does not carry any randomness; unpacking **C1** for this generator, we obtain

$$\dashv\!\!\dashv\!\!\dashv \stackrel{\mathbf{C1}}{=} \dashv\!\!\dashv\!\!\dashv$$

11:12 A Complete Diagrammatic Calculus for Conditional Gaussian Mixtures



■ **Figure 2** Axioms of CGM, the theory of CG-mixtures.

Axioms **D1**, **D2** allow us to discard any generator: in other words, if we discard the output of some generator, we might as well discard its input. In the semantics, this corresponds to the fact that all the stochastic kernels that we can represent with our syntax correspond to *normalised* (conditional) distributions.

The remaining axioms concern the interaction between the probabilistic behaviour of the discrete and the continuous parts of our syntax, mediated through the mixed *if-then-else*. Axiom **E1** captures the fact that the repeated evaluation of one condition inside of two nested if-then-else gates is redundant. Indeed, this axiom only rearranges if-then-elses, eliminating the test that depends on a shared Boolean guard, getting rid of the redundancy.

Axiom **E2** is transparent in stating the simple fact that, whenever the guard of some conditional evaluates to true (resp. false), it results on always taking the *then* (resp. *else*) branch, allowing us to discard the other one.

Axiom **E3** states a form of distributivity of if-then-else over itself, provided that we swap the corresponding guards. This corresponds to a well-known property of if-then-else gates.

Axiom **E4** is an axiom scheme which holds for any circuits c and d . It captures the intuitive idea that if the two branches of an if-then-else gate share a sample from the same Gaussian, we could also sample independently one Gaussian in each branch, since only one of them will be used. It can be seen as a form of commutativity of Gaussian sampling within a mixture-branching, allowing us to always sample locally when in presence of disjoint branches.

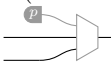
Axiom **E5** is another axiom scheme, valid for any circuit c , which captures the distributivity of all operations over if-then-else gates: evaluating a function f on two different inputs and later selecting only one of the outputs, is the same as first selecting the input and evaluating it on f . In categorical terms, if-then-else is *natural*. Note that if $c = \neg \hookrightarrow$, this axiom also allows us to copy the if-then-else gate (like **C1**, **C2** allowed us to copy some other generators), and ensures that it is a deterministic operation. Similarly, if $c = \bullet$, this axiom allows us to discard if-then-else (like **D1**, **D2** allow us to discard all other generators); in other words, we can discard all three inputs of the if-then-else generator if we have discarded its output.

Axiom **E6** encodes the simple fact that negating the guard of an if-then-else gate is equivalent to swapping the two branches.

Axiom **E7** states that a compound conditional with a conjunctive guard can be rewritten using nested if-then-else statements, where each Boolean variable is tested in sequence and the original branches are preserved accordingly.

Axiom **E8** captures the fact that if the two branches of an if-then-else are the same, then we do not need an if-then-else in the first place (and can discard the guard).

Axiom **E9** allows us to discard an if-then-else generator: similar to **D1**, **D2**, if we discard the output of some if-then-else, we might as well discard its guard and corresponding branches.

Axiom **E10** is the diagrammatic analogue of the well-known skew-associativity property of convex (aka barycentric) algebras [26]. To see this, recall (from Example 10) that the diagram  behaves as a convex sum operation. Then, **E10** allows us to re-associate different bracketings of convex sums, at the cost of changing the weights.

Crucially, our equational theory is *sound*: any syntactic equality derivable from our axioms is a valid semantic equality.

► **Theorem 12** (Soundness). *For all circuits, $c, d: v \rightarrow w$, if $c \stackrel{\text{CGM}}{=} d$ then $\llbracket c \rrbracket = \llbracket d \rrbracket$.*

Proof. It is sufficient to show that, for any of the axioms above, the two sides denote the same CG-mixture. We show that the statement holds for **E4** and **E5** – the other axioms can be checked in a similar fashion.

■ **E4:** Given that c and d have no Boolean inputs, we have

$$\llbracket c \rrbracket(x, y) = \sum_{i \in I} p_i \cdot \mathcal{N}_m \left(A_i \begin{pmatrix} x \\ y \end{pmatrix} + \mu_i, \Sigma_i \right) \quad \text{and} \quad \llbracket d \rrbracket(x, y) = \sum_{j \in J} q_j \cdot \mathcal{N}_n \left(B_j \begin{pmatrix} x \\ y \end{pmatrix} + \nu_j, \Theta_j \right)$$

Then

$$\begin{aligned} \left[\left[\begin{array}{c} \text{if-then-else gate} \\ \text{with guard } b \text{ and inputs } c, d \end{array} \right] \right] (b, x) &= \begin{cases} \sum_{i \in I} p_i \cdot \mathcal{N}_m \left(A_i \begin{pmatrix} 0 \\ y \end{pmatrix} + \mu_i, \Sigma_i \right) & \text{if } b = 1 \\ \sum_{j \in J} q_j \cdot \mathcal{N}_n \left(B_j \begin{pmatrix} 0 \\ y \end{pmatrix} + \nu_j, \Theta_j \right) & \text{if } b = 0 \end{cases} \\ &= \left[\left[\begin{array}{c} \text{if-then-else gate} \\ \text{with guard } b \text{ and inputs } c, d \end{array} \right] \right] (b, x) \end{aligned}$$

■ **E5:**

$$\begin{aligned} \left[\left[\begin{array}{c} \text{if-then-else gate} \\ \text{with guard } b \text{ and inputs } c, c \end{array} \right] \right] \left(b, \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right) &= \begin{cases} \llbracket f \rrbracket(x_1) & \text{if } b = 1 \\ \llbracket f \rrbracket(x_2) & \text{if } b = 0 \end{cases} \\ &= \begin{cases} \llbracket f \rrbracket \circ \mathcal{N}_n(x_1, 0) & \text{if } b = 1 \\ \llbracket f \rrbracket \circ \mathcal{N}_n(x_2, 0) & \text{if } b = 0 \end{cases} = \left[\left[\begin{array}{c} \text{if-then-else gate} \\ \text{with guard } b \text{ and input } c \end{array} \right] \right] \left(b, \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right) \blacktriangleleft \end{aligned}$$

6 Completeness

The main contribution of our work is the converse of Theorem 12, which states the *completeness* of our theory for the chosen semantics.

Our proof of completeness follows a normal form argument, where a syntactic representative is given for every semantic object in the image of the interpretation functor considered, later showing that when two circuits have the same semantics, they can always be transformed into their unique syntactic representative through a series of rewrite steps, following the equational axioms presented in Section 5. The core of the completeness argument is thus a normalisation proof: an explicit procedure to rewrite any given circuit into a normal form using only the axioms of our equational theory.

At a high-level, our completeness proof consists of two main steps, of which the first one is the most technically challenging.

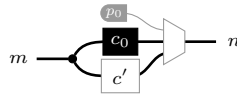
First, we explain how to normalise circuits of type $w \rightarrow \mathbb{R}^n$, *i.e.*, where the only possible outputs are real-valued random variables. Semantically, these correspond to mixtures of Gaussians (conditioned on their inputs, which can be both discrete or continuous). At a high level, the normalisation proof proceeds by structural induction: assuming that we have some circuit d in normal form, we show how to normalise any circuit obtained by composing d with any of the generators in our signature.

Second, we address the normalisation of arbitrary circuits – which can have both Boolean and real outputs – by decomposing them into two parts: a purely Boolean subcircuit and another subcircuit of type $w \rightarrow \mathbb{R}^n$. Finally, the completeness for Boolean circuits and the previous normalisation procedure for circuits of type $w \rightarrow \mathbb{R}^n$ will allow us to obtain the general completeness result we are looking for.

► **Remark 13.** In what follows we will focus on circuits of type $\mathbb{B}^p \mathbb{R}^m \rightarrow \mathbb{B}^q \mathbb{R}^n$ without loss of generality. Indeed, for any arbitrary circuit $d: v \rightarrow w$, we can always pre and post-compose it with the wire crossings (\times , \times , \times , \times) to move all wires of type \mathbb{B} before those of type \mathbb{R} (while preserving the order within the wires of each type), thereby obtaining a circuit $R(d): \mathbb{B}^p \mathbb{R}^m \rightarrow \mathbb{B}^q \mathbb{R}^n$. This process is clearly reversible: it suffices to pre and post-compose with the same symmetries in reverse to obtain the circuit d with which we started. Moreover, for any two circuits $c, d: \mathbb{B}^p \mathbb{R}^m \rightarrow \mathbb{B}^q \mathbb{R}^n$, $c = d$ if and only if $R(c) = R(d)$. This is because, using the laws of SMCs, any axiom that we use to show that $c = d$ can be applied to show that $R(c) = R(d)$ and vice-versa.

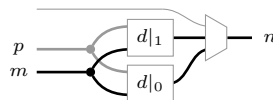
For clarity, and to simplify the overall proof, we break the definition of our normal form into parts. This highlights the interaction between the different components and allows us to define different normalisation procedures for circuits with different types, depending on whether they have Boolean inputs or outputs. First, the normal form for circuits with no Boolean inputs or outputs is a simple cascade of convex sums.

► **Definition 14.** A circuit $c: \mathbb{R}^m \rightarrow \mathbb{R}^n$ is in convex normal form (CNF) if it is in the form



where $p_0 \in (0, 1)$, $c_0: \mathbb{R}^m \rightarrow \mathbb{R}^n$ is in GaussCirc, and $c': \mathbb{R}^m \rightarrow \mathbb{R}^n$ is in CNF or in GaussCirc.

► **Definition 15.** A circuit $d: \mathbb{B}^{p+1} \mathbb{R}^m \rightarrow \mathbb{R}^n$ is in normal form (NF) if it is in the form inductively defined below,



where $d|_i, i \in \{0, 1\}$ are themselves in normal form, or in CNF (base case, see Definition 14).

Intuitively, $d|_1$ corresponds with the circuit that results from d conditioned on its first Boolean variable being true, and similarly with $d|_0$, where the same Boolean variable is assumed to be false instead. The uniqueness of these normal forms stems from the unique characterisation of CG-mixtures by their parameters (Proposition 5).

► **Proposition 16.** *Any two circuits $c, d: \mathbb{B}^p \mathbb{R}^m \rightarrow \mathbb{R}^n$ in NF such that $\llbracket c \rrbracket = \llbracket d \rrbracket$, are equal.*

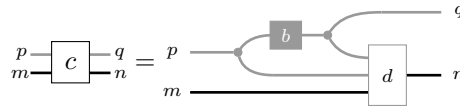
Intuitively, our normal form represents a circuit as a *conditional probability distribution tree* [11, Chapter 5] – a rooted tree where leaf nodes are labelled with distributions and interior nodes correspond to parent variables, with each outgoing edge associated with a unique variable assignment. This global structure, which captures the entire distribution, is mirrored by our circuits: convex combinations of (multivariate) Gaussians take the role of the leaves, and if-then-else gates “split” the tree by conditioning on discrete variables.

The axioms of CGM are sufficient to rewrite any given circuit without any Boolean outputs into one in normal form. The proof of this is the main technical contribution of our paper, and relies on a lengthy structural induction.

► **Theorem 17.** *Any circuit of type $\mathbb{B}^p \mathbb{R}^m \rightarrow \mathbb{R}^n$ is equal to one in normal form.*

The normal form for arbitrary circuits below is a diagrammatic form of *disintegration*, mirroring how a joint distribution $p(x, y)$ can be decomposed (that is, disintegrated) into the product $p(x)p(y|x)$ of a marginal and a conditional distribution. Any circuit c can be decomposed in the same way as the composition of a Boolean circuit b and a mixed circuit d with only real-valued output. Semantically, the $\llbracket d \rrbracket$ is the CG-mixture obtained by conditioning $\llbracket c \rrbracket$ on its Boolean output.

► **Definition 18.** *A circuit $c: \mathbb{B}^p \mathbb{R}^m \rightarrow \mathbb{B}^q \mathbb{R}^n$ is in normal form (NF) if there exists some Boolean circuit b and some mixed circuit d in normal form (Definition 15) such that*



which satisfy $\llbracket d \rrbracket (\cdot | a', a, x) = \mathcal{N}_n(0, 0)$ if $\llbracket b \rrbracket (a' | a) = 0$, for $a \in \mathbb{B}^p, a' \in \mathbb{B}^q$, and $x \in \mathbb{R}^m$.

The last condition is here to deal with edge cases in which we are conditioning on events of measure zero – a classic issue when defining disintegrations. When $\llbracket b \rrbracket (a' | a) = 0$, the conditional $\llbracket d \rrbracket$ is not well-defined when its Boolean input is (a', a) ; therefore, any circuit will do to represent this case. However, to guarantee the uniqueness of the normal form, we need a convention: here, we choose $\mathcal{N}_n(0, 0)$ or, equivalently, a Dirac concentrated at $0 \in \mathbb{R}^n$.

► **Proposition 19.** *Normal forms are unique, i.e., for any two circuits $c, c': \mathbb{B}^p \mathbb{R}^m \rightarrow \mathbb{B}^q \mathbb{R}^n$ in normal form, if $\llbracket c \rrbracket = \llbracket c' \rrbracket$ then $c \stackrel{\text{CGM}}{=} c'$.*

The following theorem relies extensively on Theorem 17 (normalisation of circuits without Boolean outputs); in this more general case, we simply need to check that the side condition of Definition 18 (guaranteeing the uniqueness of the normal form) is satisfied.

► **Theorem 20.** *Every circuit $\mathbb{B}^p \mathbb{R}^m \rightarrow \mathbb{B}^q \mathbb{R}^n$ is equal to one in normal form.*

Putting together the last two results, we can prove the completeness of our theory.

► **Theorem 21 (Completeness).** *For any two circuits c, d , if $\llbracket c \rrbracket = \llbracket d \rrbracket$ then $c \stackrel{\text{CGM}}{=} d$.*

Proof. By Remark 13, we can assume wlog that c, d are two circuits of type $\mathbb{B}^p \mathbb{R}^m \rightarrow \mathbb{B}^q \mathbb{R}^n$. By Theorem 20, these are equal to circuits c' and d' in normal form. Then, by soundness, if $\llbracket c \rrbracket = \llbracket d \rrbracket$, we also have $\llbracket c' \rrbracket = \llbracket d' \rrbracket$ and therefore $c' \stackrel{\text{CGM}}{=} d'$, by Proposition 19, since both circuits are in normal form. Finally, by transitivity of equality, $c \stackrel{\text{CGM}}{=} d$. ◀

7 Conclusions

We presented a sound and complete axiomatisation of equivalence for CG-mixtures. We achieved this result through a calculus of string diagrams, whose semantics target morphisms in `MixGauss`, a suitable category that combines discrete and Gaussian probability, as well as the complex interactions between them.

While PGMs have long provided a flexible framework for specifying decision systems that operate in environments characterised by uncertainty [1, 11, 12, 19, 20], it is only in recent work that the notion of Markov categories [6] and its diagrammatic syntax have been applied to the formalisation of these graphical models [5, 10, 9, 16]. Our work extends this line of research by giving a diagrammatic equational theory capable of capturing CG-mixtures, providing compositional semantics and a detailed image of the internal construction of the distribution represented.

We plan to extend this work in two orthogonal directions. First, we aim to extend our diagrammatic calculus with a primitive for *conditioning*, enabling us to incorporate observed data into our models. An axiomatisation for CG-mixtures with conditioning would allow us to implement inference – particularly, parameter learning – as equational reasoning. Second, we plan to generalise our results to a broader classes of mixture models. Indeed, the only property of Gaussians on which our completeness proof depends is that mixtures of Gaussians are uniquely determined by their parameters (Proposition 3). This property underpins our definition of the normal form we use to show completeness. The rest of the proof is modular and can be adapted to other subcategories of stochastic maps, such as exponential or Poisson distributions, provided that similar characterisation results hold for their mixtures.

References

- 1 J. M. Bernardo and A. Smith. *Bayesian Theory*. Wiley series in probability and statistics. John Wiley & Sons Ltd., Chichester, 2000.
- 2 Patrick Billingsley. *Probability and measure*. A Wiley-Interscience publication. Wiley, New York [u.a.], 3. ed edition, 1995.
- 3 Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.
- 4 Filippo Bonchi, Robin Piedeleu, Paweł Sobociński, and Fabio Zanasi. Graphical affine algebra. In *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '19*. IEEE Press, 2021.
- 5 Brendan Fong. Causal theories: A categorical perspective on Bayesian networks, 2013. [arXiv:1301.6201](https://arxiv.org/abs/1301.6201).
- 6 Tobias Fritz. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370:107239, August 2020.
- 7 Tobias Fritz and Andreas Klingler. The d-separation criterion in categorical probability. *Journal of Machine Learning Research*, 24(46):1–49, 2023. URL: <https://jmlr.org/papers/v24/fritz22-0916.html>.
- 8 Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: <http://www.deeplearningbook.org>.
- 9 Bart Jacobs, Aleks Kissinger, and Fabio Zanasi. Causal inference by string diagram surgery. In *International conference on foundations of software science and computation structures*, pages 313–329. Springer, 2019. doi:10.1007/978-3-030-17127-8_18.
- 10 Bart Jacobs, Fabio Zanasi, et al. The logical essentials of bayesian reasoning. *Foundations of Probabilistic Programming*, pages 295–332, 2020. doi:10.1017/9781108770750.010.
- 11 D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. Adaptive computation and machine learning. MIT Press, 2009.

- 12 Steffen L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.
- 13 Steffen L. Lauritzen and Frank Jensen. Stable local computation with conditional Gaussian distributions. *Statistics and Computing*, 11(2):191–203, April 2001. doi:10.1023/A:1008935617754.
- 14 Steffen L Lauritzen and Nanny Wermuth. *Mixed interaction models*. Institut for Elektroniske Systemer, Aalborg Universitetscenter, 1984.
- 15 Steffen L Lauritzen and Nanny Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *The annals of Statistics*, pages 31–57, 1989.
- 16 Antonio Lorenzin and Fabio Zanasi. An algebraic approach to moralisation and triangulation of probabilistic graphical models, 2025. doi:10.48550/arXiv.2503.11820.
- 17 Saunders MacLane. *Categories for the Working Mathematician*. Springer New York, 1971. doi:10.1007/978-1-4612-9839-7.
- 18 Geoffrey J McLachlan and David Peel. *Finite Mixture Models*. Wiley Series in Probability and Statistics. John Wiley & Sons, Nashville, TN, September 2000.
- 19 Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
- 20 Judea Pearl. *Causality*. Cambridge University Press, 2 edition, 2009.
- 21 Robin Piedeleu, Mateo Torres-Ruiz, Alexandra Silva, and Fabio Zanasi. A complete axiomatisation of equivalence for discrete probabilistic programming. In *European Symposium on Programming*, pages 202–229. Springer, 2025. doi:10.1007/978-3-031-91121-7_9.
- 22 Robin Piedeleu and Fabio Zanasi. *An Introduction to String Diagrams for Computer Scientists*. Cambridge University Press, 2025.
- 23 C. Radhakrishna Rao. *Linear Statistical Inference and its Applications: Second Editon*. Wiley, April 1973.
- 24 P. Selinger. *A Survey of Graphical Languages for Monoidal Categories*, pages 289–355. Springer Berlin Heidelberg, 2010.
- 25 Dario Stein, Fabio Zanasi, Robin Piedeleu, and Richard Samuelson. Graphical quadratic algebra. In *International Colloquium on Theoretical Aspects of Computing*, pages 298–316. Springer, 2025.
- 26 Marshall Harvey Stone. Postulates for the barycentric calculus. *Annali di Matematica Pura ed Applicata*, 29:25–30, 1949.
- 27 Hsi Guang Sung. *Gaussian mixture regression and classification*. Rice University, 2004.
- 28 Mateo Torres-Ruiz, Robin Piedeleu, Alexandra Silva, and Fabio Zanasi. A complete diagrammatic calculus for conditional gaussian mixtures, 2025. doi:10.48550/arXiv.2510.04649.