Probabilistic Circuits for Cumulative Distribution Functions

Oliver Broadrick[1] **William Cao**[1] **Benjie Wang**[1] **Martin Trapp**[2] **Guy Van den Broeck**[1]

[1]Computer Science Dept., University of California , Los Angeles, California, USA
[2]Department of Computer Science, Aalto University, Espoo, Finland

Abstract

A probabilistic circuit (PC) succinctly expresses a function that represents a multivariate probability distribution and, given sufficient structural properties of the circuit, supports efficient probabilistic inference. Typically a PC computes the probability mass (or density) function (PMF or PDF) of the distribution. We consider PCs instead computing the cumulative distribution function (CDF). We show that for distributions over binary random variables these representations (PMF and CDF) are essentially equivalent, in the sense that one can be transformed to the other in polynomial time. We then show how a similar equivalence holds for distributions over finite discrete variables using a modification of the standard encoding with binary variables that aligns with the CDF semantics. Finally we show that for continuous variables, smooth, decomposable PCs computing PDFs and CDFs can be efficiently transformed to each other by modifying only the leaves of the circuit.

1 INTRODUCTION

Modeling multivariate probability distributions in a way that is both expressive and allows efficient probabilistic reasoning is a fundamental problem in the field of artificial intelligence. Probabilistic circuits (PCs) provide a unifying framework for a myriad of tractable probabilistic models and reduce tractability to syntactic properties of the underlying circuit [Darwiche, 2003, Poon and Domingos, 2011, Choi et al., 2020]. In general, a PC computes a multilinear polynomial in its inputs, most commonly computing a probability mass (or density) function (PMF/PDF). However, other polynomials can be used to encode probability distributions and have been studied as alternative semantics for PCs, including generating functions [Zhang et al., 2021, Harviainen et al., 2023, Bläser, 2023] and characteristic functions [Yu et al., 2023]. In this paper, we extend this line of work and consider the *cumulative distribution function* (CDF) as a semantics for PCs.

Unlike the PMF, which computes the probability of an input assignment, the CDF computes the probability of realizing any assignment with entries elementwise less than or equal to the input assignment. The CDF exists for every real-valued multivariate probability distribution, which is not true for mass or density functions, and has broad applications in machine learning and statistics [Huang, 2009, Huang and Jojic, 2010, Hyvärinen and Pajunen, 1999, Gresele et al., 2021]. A particularly useful property of the CDF arises from its interpretation as a transformation, i.e., CDF transformed random variables are uniformly distributed. This property has been heavily exploited in the literature, for example, in copulas [Ling et al., 2020], for inverse transform sampling [Gentle, 2003], and density estimation via boosting [Awaya and Ma, 2021].

While CDFs are defined for arbitrary distributions over real-valued random variables, we consider three important special cases. We begin with distributions over binary random variables for which PCs are known to be tractable when they express a multilinear polynomial that computes the PMF (called the *PMF polynomial*). Therefore we consider multilinear polynomials that compute the CDF, which we call *CDF polynomials*. We find that for binary random variables, CDF polynomials are exactly equal to probability generating functions (PGFs). This surprising equivalence immediately implies that a circuit computing the PMF (respectively CDF) can be transformed to a circuit computing the CDF (respectively PMF) in polynomial time, by recent results [Broadrick et al., 2024, Agarwal and Bläser, 2024]. Moreover, our new interpretation of PGFs as CDFs enables us to give an alternative and interesting proof of the transformation from PGFs to PMFs based on the generalized principle of inclusion-exclusion.

Next we consider distributions over finite discrete random

Accepted for the 7th Workshop on Tractable Probabilistic Modeling at UAI (TPM 2024).

variables. Typically in the PC literature, such variables are handled by encoding them with binary variables, e.g., with a one-hot encoding. We find that by using a simple new encoding that respects the less-than-or-equal-to relation, we can ensure not only that the PMFs agree up to the encoding (as normal) but that also the CDFs agree up to the encoding. This allows us to reduce the finite discrete case to the binary case, applying the transformations for the binary case to obtain a similar equivalence.

Finally, we consider continuous variables, finding that for PCs which are smooth and decomposable – standard structural properties used in the literature [Darwiche and Marquis, 2002, Choi et al., 2020] – simple modifications to the leaves of the PC enable transformations between PDFs and CDFs. Our results relating CDFs to PMFs/PDFs complement and extend those of Broadrick et al. [2024] and Agarwal and Bläser [2024], who find that several other PC semantics are equivalent for binary random variables but do not consider CDFs.

2 BACKGROUND ON CIRCUITS

We study probabilistic circuits (PCs): computation graphs that, given sufficient structural properties, render inference tasks tractable. Let $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ be random variables, and denote the set of all assignments to $\boldsymbol{X}$ by $\mathrm{val}(\boldsymbol{X})$.

Definition 1. *A* probabilistic circuit *(PC) in variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ is a rooted directed acyclic graph. Each node v is either (i) a product node, (ii) a sum node with edges to children labeled by weights $w_{v1}, \ldots, w_{vk} \in \mathbb{R}$, or (iii) a leaf node, labeled by a function $l_v : val(X_i) \to \mathbb{R}$ for some X_i. Each node v (with children $v_1, \ldots, v_k$) computes a polynomial whose indeterminates are $l_1, \ldots, l_m$ (the functions labeling the leaves of the PC):*

$$p_v(l_1, \ldots, l_m) = \begin{cases} \prod_{i=1}^k p_{v_i}(l_1, \ldots, l_m) & \text{if } v \text{ a product node} \\ \sum_{i=1}^k w_{vi} p_{v_i}(l_1, \ldots, l_m) & \text{if } v \text{ a sum node} \\ l_v & \text{if } v \text{ a leaf node.} \end{cases}$$

The polynomial p computed by a PC is the polynomial computed by its root. The function computed by a PC is $P : val(\boldsymbol{X}) \to \mathbb{R}$ given by

$$P(\boldsymbol{x}) = p(l_1(\boldsymbol{x}), \ldots, l_m(\boldsymbol{x})).$$

Lastly, the size of a PC is the number of edges in it.

If each X_i is binary, taking values in $\{0, 1\}$, we commonly consider the leaf functions x_i and $\bar{x}_i$ respectively mapping the bit $b \in \{0, 1\} = \mathrm{val}(X_i)$ to b and $1 - b$. In the case that we only use the leaf functions x_i, then the polynomial computed by a PC and the function computed by a PC effectively coincide.

Note that we will typically assume the functions labeling the leaves, sometimes called *input functions*, are tractable, meaning that arbitrary integrals and derivatives can be computed efficiently. We also note that the polynomials considered in this paper are *multilinear*, meaning that they are linear in each variable. For example, the polynomials $x_1x_3 - x_2x_3$ and $x_1x_2x_3 + 1$ are multilinear, but x^2 and $x_1x_2^7 + 1$ are not. For the remainder of this paper we will use circuit to mean PC.

3 CUMULATIVE DISTRIBUTION FUNCTIONS

For real-valued random variables $X_1, \ldots, X_n$ the *cumulative distribution function* (CDF) $F : \mathbb{R}^n \to [0, 1]$ is

$$F(x_1, \ldots, x_n) = \mathbb{P}[X_1 \leq x_1, \ldots, X_n \leq x_n].$$

This is a general notion; the CDF exists, e.g., regardless of whether the variables are discrete, continuous, or otherwise. If the random variables are all discrete, meaning that they take values in some countable subset of the reals $D \subset \mathbb{R}$, then the distribution can also be specified by a *probability mass function* (PMF) $f : D^n \to [0, 1]$ given by

$$f(x_1, \ldots, x_n) = \mathbb{P}[X_1 = x_1, \ldots, X_n = x_n].$$

If instead all the random variables are absolutely continuous, then there exists a *probability density function* (PDF) $f : \mathbb{R}^n \to \mathbb{R}^{\geq 0}$ given by

$$f(x_1, \ldots, x_n) = \frac{\partial^n}{\partial x_1 \ldots \partial x_n} F(x_1, \ldots, x_n).$$

In the following sections we consider three cases. The first two concern discrete distributions: those over binary random variables in Section 4, and those over finite discrete variables in Section 5. We then consider continuous variables in Section 6.

4 BINARY VARIABLES

We first consider the simplest setting: probability distributions over binary random variables. Let $X_1, \ldots, X_n$ be random variables taking values in $\{0, 1\} \subset \mathbb{R}$. Then there exists a PMF $f : \{0, 1\}^n \to [0, 1]$ with $f(x_1, \ldots, x_n) = \mathbb{P}[X_1 = x_1, \ldots, X_n = x_n]$. Moreover, there is a unique multilinear polynomial $p \in \mathbb{R}[x_1, \ldots, x_n]$ that computes f (in the sense that it agrees with f on all inputs in $\{0, 1\}^n$) which we will call the *PMF polynomial*:

$$p(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} f(v_S) \prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i) \quad (1)$$

where $[n] = \{1, \ldots, n\}$ and $v_S \in \{0, 1\}^n$ is the characteristic vector of S (v_S has ith entry 1 if $i \in S$ and ith entry

X_1	X_2	f	F
0	0	.1	.1
0	1	.4	.5
1	0	.2	.3
1	1	.3	1.0

$$p(x_1, x_2) = -.2x_1x_2 + .1x_1 + .3x_2 + .1 \qquad \text{(Eq. (1))}$$
$$c(x_1, x_2) = .3x_1x_2 + .2x_1 + .4x_2 + .1 \qquad \text{(Eq. (2))}$$

Figure 1: A probability distribution over two binary random variables. The PMF and CDF functions are specified in the table, and the corresponding PMF and CDF polynomials p and c are given below.

0 if $i \notin S$). As an example, consider the PMF f with PMF polynomial p given in Fig. 1.

Like the PMF, the CDF can also be uniquely expressed as a multilinear polynomial $c \in \mathbb{R}[x_1, \ldots, x_n]$ which we call the *CDF polynomial*:

$$c(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} F(v_S) \prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i). \qquad (2)$$

Note that p and c can be related simply:

$$c(x) = \sum_{y \leq x} p(x) \qquad (3)$$

where $y \leq x$ is elementwise, meaning that $y_i \leq x_i$ for every i. Again consider the example in Fig. 1 with CDF F and CDF polynomial c.

To understand how circuits computing PMF and CDF polynomials relate, we recall one additional polynomial representation for probability distributions, the *probability generating function* (PGF). For binary random variables, this is the polynomial

$$g(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} f(v_S) \prod_{i \in S} x_i \qquad (4)$$

where $v_S \in \{0, 1\}^n$ is the characteristic vector of S. We now make a perhaps surprising and satisfying observation.

Proposition 1. *Fix a probability distribution over binary random variables $X_1, \ldots, X_n$ taking values $\{0, 1\} \subset \mathbb{R}$ with CDF polynomial c and PGF g. Then $c = g$.*

Proof. While c and g are polynomials, we abuse notation and use the same names to refer to the functions $c, g : \{0, 1\}^n \to \mathbb{R}$ that they induce. Observe then that

$$\begin{aligned} g(x) &= \sum_{S \subseteq [n]} f(v_S) \prod_{i \in S} x_i \\ &= \sum_{y \leq 1^n} f(y) \prod_{i : y_i = 1} x_i \\ &= \sum_{y \leq x} f(y) = c(x). \end{aligned}$$

Here the first equality follows from definition (Eq. 4), the second equality follows by identifying sets with their characteristic vectors, the third equality holds because if $y > x$ (elementwise) then $\prod_{i \in S} x_i = 0$, and the final equality follows from Eq. 3. Finally, the equality of the functions implies equality of the polynomials. $\square$

Proposition 1 immediately implies that circuits computing p (respectively c) can be transformed to circuits computing c (respectively p) in polynomial time[1] by the results of Broadrick et al. [2024] and Agarwal and Bläser [2024]. In other words, for binary variables circuits computing PMF polynomials and circuits computing CDF polynomials are equally succinct probabilistic models.

Corollary 1. *Fix a probability distribution over n binary random variables with PMF polynomial p and CDF polynomial c. A circuit of size s computing p (respectively c) can be transformed to a circuit computing c (respectively p) in time $O(ns)$.*

While Broadrick et al. [2024] and Agarwal and Bläser [2024] already show how to transform a circuit computing a PGF to a circuit computing a PMF, Proposition 1 allows us to give an alternative proof, which, while essentially equivalent, provides a satisfying explanation for what the transformation is doing. Specifically, our interpretation of the PGF as a CDF allows us to view the transformation as an efficient application of the generalized principle of inclusion-exclusion. We provide the proof in Appendix A.

5 FINITE DISCRETE VARIABLES

We now consider distributions over finite discrete random variables, those taking values in some finite set of reals, $K \subset \mathbb{R}$. In particular, we focus on the case $K = \{0, 1, \ldots, k-1\}$ for some $k \in \mathbb{N}$. We note that any other finite set of reals of size k may be bijected with this set, as well as other categorical or ordinal random variables.

The standard approach in the probabilistic circuits literature for handling such finite discrete variables is to encode them

[1]The complexity bound given in Corollary 1 is improved by a factor of n compared to that presented in [Broadrick et al., 2024]; this improvement follows from the use of more efficient homogenization based on polynomial interpolation [Saptharishi, 2015, Lemma 5.4].

with binary indicators, typically using a one-hot encoding. In particular, a distribution $\mathbb{P}$ over K-valued random variables is mapped to a distribution $\mathbb{P}'$ over binary random variables such that the two distributions are equivalent up to an injective mapping $\phi : K^n \to \{0,1\}^{kn}$. That is, for all $x \in K^n$, we have $\mathbb{P}(x) = \mathbb{P}'(\phi(x))$. While the PMFs agree up to ϕ, the same cannot be said in general for the CDFs. While both CDFs are sums over their respective PMFs, the terms in each of these sums will not in general be the same up to ϕ (for example, with a one-hot encoding even in the univariate case, $2 \leq 3$ but $(0,1,0) \not\leq (0,0,1)$).

However, we observe that there is a simple alternative encoding of finite discrete variables with binary variables that does respect the elementwise less-than-or-equal-to relation.

Definition 2. *(Less-Than Encoding) For $k \in \mathbb{N}$, the Less-Than Encoding is the function $LT_k : \{0,1,\dots,k-1\} \to \{0,1\}^k$ given by*

$$\mathrm{LT}_k(x) := \begin{bmatrix} \mathbf{1}_x \\ \mathbf{0}_{k-x} \end{bmatrix}$$

with $\mathbf{1}_n = (1,\dots,1)^\top \in \mathbb{R}^n$ and $\mathbf{0}_n = (0,\dots,0)^\top \in \mathbb{R}^n$ respectively.

Note that the Less-Than Encoding respects the less-than-or-equal-to relation in the sense that for any $x, y \in \{0,1,\dots,k-1\}$, $x \leq y$ if and only if $\mathrm{LT}_k(x) \leq \mathrm{LT}_k(y)$.

We are now free to map a distribution $\mathbb{P}$ over K-valued random variables to a distribution $\mathbb{P}'$ over binary random variables using the Less-Than Encoding and preserving not only

$$p(x) = p'(\mathrm{LT}_k(x))$$

but also

$$c(x) = \sum_{y \leq x} p(y) = \sum_{y \leq \mathrm{LT}_k(x)} p'(y) = c(\mathrm{LT}_k(x)).$$

In particular, Corollary 1 can be applied to distributions over finite discrete variables using this encoding.

6 CONTINUOUS VARIABLES

Lastly we consider the case where all of the variables are continuous with the distribution admitting a joint density function. Let $X_1, \dots, X_n$ be real-valued random variables with CDF $F : \mathbb{R}^n \to [0,1]$ and PDF $f : \mathbb{R}^n \to \mathbb{R}^{\geq 0}$. In this setting, we show that it is straightforward to efficiently transform between circuits computing the PDF and CDF *when the circuits are smooth and decomposable*. Smoothness and decomposability are standard structural properties of PCs that enable efficient inference [Darwiche and Marquis, 2002, Choi et al., 2020]. In order to define them, we use the notion of the *scope* of a node v denoted scope(v) which is the set of all variables appearing in the sub-PC rooted at v.

Definition 3 (Smoothness)**.** *A sum node v with children $v_1, \dots, v_k$ is smooth if the scope of its children are equal to its own scope: scope(v) = scope(v_i) for $i = 1, \dots, k$.*

Definition 4 (Decomposable)**.** *A product node v with children v_1 and v_2 is decomposable if the scope of its children partition its scope: scope(v) = scope$(v_1) \cup$ scope(v_2) and scope$(v_1) \cap$ scope$(v_2) = \emptyset$.*

It is well known that a smooth and decomposable PC supports efficient integration, by pushing integrations to the leaves [Choi et al., 2020]. In particular, the following proposition follows by simply replacing each input distribution $l(x)$ with a new input distribution $l_{CDF}(x) = \int_{-\infty}^{x} l(t)dt$.

Proposition 2. *A smooth, decomposable PC of size s computing the PDF f can be transformed to a smooth, decomposable PC of size s computing the CDF F.*

In order to perform a transformation in the other direction (i.e., from CDF to PDF), decomposability alone suffices. Intuitively, partial derivatives can be pushed over sum nodes by linearity and can be pushed over product nodes because decomposability ensures that one of the two terms in the product rule is zero. A proof of the following claim is given in Appendix A.

Proposition 3. *A decomposable PC of size s computing the CDF F can be transformed to a decomposable PC of size s computing the PDF f.*

7 CONCLUSION

We study a basic question: what if a tractable PC computes a CDF instead of a PMF or PDF? We show that in three important cases the two models are roughly equivalent. For distributions over binary random variables, we observe that the CDF polynomial is exactly the probability generating function (PGF) of the distribution. This observation allows us to conclude that PCs computing a PMF polynomial or CDF polynomial can be transformed to each other in polynomial time, and we were able to give a new explanation for how a circuit computing the PGF can be transformed to a circuit computing the PMF. We then showed how to reduce the finite discrete case to the binary case with a slight modification to the standard encoding with binary indicators that respects the less-than-or-equal-to relation needed to compute the CDF. Finally, we showed how the standard structural properties of smoothness and decomposability suffice in the continuous case to make transformations between PDFs and CDFs simplify to modifications of the leaves. We leave open the question of whether similar results hold for circuit models handling discrete variables over infinite domains, mixed distributions, or finite discrete variables with more efficient encodings into binary variables, e.g. like those used by Cao et al. [2023] for finite discrete variables or those obtained by Garg et al. [2023] for discretizations of continuous densities.

Acknowledgements

We thank Sanyam Agarwal for referring us to [Saptharishi, 2015].

References

Sanyam Agarwal and Markus Bläser. Probabilistic generating circuits – demystified, 2024.

Naoki Awaya and Li Ma. Unsupervised tree boosting for learning probability distributions. *arXiv preprint arXiv:2101.11083*, 2021.

Markus Bläser. Not all strongly rayleigh distributions have small probabilistic generating circuits. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2592–2602. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/blaser23a.html`.

Oliver Broadrick, Honghua Zhang, and Guy Van den Broeck. Polynomial semantics of tractable probabilistic circuits. In *40th Conference on Uncertainty in Artificial Intelligence*, 2024.

Richard A Brualdi. *Introductory combinatorics*. Pearson Education India, 2004.

William X Cao, Poorva Garg, Ryan Tjoa, Steven Holtzen, Todd Millstein, and Guy Van den Broeck. Scaling integer arithmetic in probabilistic programs. In *Uncertainty in Artificial Intelligence*, pages 260–270. PMLR, 2023.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. oct 2020. URL `http://starai.cs.ucla.edu/papers/ProbCirc20.pdf`.

Adnan Darwiche. A differential approach to inference in bayesian networks. *J. ACM*, 50(3):280–305, may 2003. ISSN 0004-5411. doi: 10.1145/765568.765570. URL `https://doi.org/10.1145/765568.765570`.

Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

Poorva Garg, Steven Holtzen, Guy Van den Broeck, and Todd Millstein. Bit blasting probabilistic programs. *arXiv preprint arXiv:2312.05706*, 2023.

James E Gentle. *Random number generation and Monte Carlo methods*. Springer, 2003.

Luigi Gresele, Julius Von Kügelgen, Vincent Stimper, Bernhard Schölkopf, and Michel Besserve. Independent mechanism analysis, a new concept? *Advances in neural information processing systems*, 34:28233–28248, 2021.

Juha Harviainen, Vaidyanathan Peruvemba Ramaswamy, and Mikko Koivisto. On inference and learning with probabilistic generating circuits. In *The 39th Conference on Uncertainty in Artificial Intelligence*, 2023.

Jim Huang and Nebojsa Jojic. Maximum-likelihood learning of cumulative distribution functions on graphs. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 342–349. JMLR Workshop and Conference Proceedings, 2010.

Jim C Huang. *Cumulative distribution networks: Inference, estimation and applications of graphical models for cumulative distribution functions*. University of Toronto Toronto, ON, Canada, 2009.

Aapo Hyvärinen and Petteri Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12 (3):429–439, 1999. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(98)00140-3. URL `https://www.sciencedirect.com/science/article/pii/S0893608098001403`.

Chun Kai Ling, Fei Fang, and J. Zico Kolter. Deep archimedean copulas. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1535–1545. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/10eb6500bd1e4a3704818012a1593cc3-Paper.pdf`.

Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–690, 2011. doi: 10.1109/ICCVW.2011.6130310.

Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. *Github survey*, 95, 2015.

Zhongjie Yu, Martin Trapp, and Kristian Kersting. Characteristic circuit. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Honghua Zhang, Brendan Juba, and Guy Van den Broeck. Probabilistic generating circuits. In *International Conference on Machine Learning*, pages 12447–12457. PMLR, 2021.

Probabilistic Circuits for Cumulative Distribution Functions (Supplementary Material)

Oliver Broadrick[1] **William Cao**[1] **Benjie Wang**[1] **Martin Trapp**[2] **Guy Van den Broeck**[1]

[1]Computer Science Dept., University of California , Los Angeles, California, USA
[2]Department of Computer Science, Aalto University, Espoo, Finland

A PROOFS

We first give an alternative proof that, for distributions over binary random variables, a circuit computing the PGF can be transformed to a circuit computing the PMF in polynomial time.

We identify vectors $x \in \{0,1\}^n$ with sets $S_x = \{i : x_i = 1\}$ and then view p and c as set functions, $p, c : \mathcal{P}([n]) \to \mathbb{R}$ where $\mathcal{P}([n])$ is the power set of $[n]$, yielding

$$c(S) = \sum_{T \subseteq S} p(T) \tag{5}$$

from Equation 3. We now recall the generalized principle of inclusion-exclusion, which allows 'inversion' of set-functions of the form in Eq. (5) (see e.g. Brualdi [2004]).

Fact 1 (Inclusion-Exclusion)**.** *For a finite set S, let $f : \mathcal{P}(S) \to \mathbb{R}$ be an arbitrary function (where $\mathcal{P}(S)$ is the power set of S), and let $g : \mathcal{P}(S) \to \mathbb{R}$ be given by:*

$$g(A) = \sum_{B \subseteq A} f(B).$$

Then,

$$f(A) = \sum_{B \subseteq A} (-1)^{|A|-|B|} g(B).$$

Therefore, applying Fact 1 to Eq. (5), we obtain

$$p(S) = \sum_{T \subseteq S} (-1)^{|S|-|T|} c(T). \tag{6}$$

While this successfully expresses p in terms of c, it also introduces a sum over exponentially many terms, and so any direct construction of a circuit based on this expression yields a circuit of exponential size. However it is possible to use a certain form of c to compute Eq. (6) in a single forward pass (and to construct a circuit for c).

For any multilinear polynomial

$$f(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} \alpha_S \prod_{i \in S} x_i$$

with $\alpha_S \in \mathbb{R}$, we define the *network*[1] form of f as

$$\bar{f}(x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n) = \sum_{S \subseteq [n]} f(v_S) \prod_{i \in S} x_i \prod_{i \notin S} \bar{x}_i$$

[1]Here the term network refers to such polynomials' origin in Bayesian Network inference [Darwiche, 2003].

where $v_S \in \{0,1\}^n$ has $x_i = 1$ for $i \in S$ and $x_i = 0$ for $i \notin S$. Note that if we have a circuit computing $\bar{f}$, then a circuit computing f can be obtained easily by replacing each $\bar{x}_i$ with $1 - x_i$. The following lemma allows us to also efficiently transform a circuit computing f to a circuit computing $\bar{f}$.[2]

Lemma 1 (Broadrick et al. [2024]). *Given a circuit of size s computing multilinear polynomial f, a circuit computing $\bar{f}$ can be constructed in time $O(ns)$.*

Now, back to the problem of transforming a circuit computing c to one computing p, we first apply Lemma 1 to obtain a circuit computing

$$\bar{c}(x, \bar{x}) = \sum_{S \subseteq [n]} c(v_S) \prod_{i \in S} x_i \prod_{i \notin S} \bar{x}_i. \tag{7}$$

We now observe that we can compute $p(x)$ using Eq. (6) by evaluating $\bar{c}(y, \bar{y})$ for a carefully crafted input $(y, \bar{y})$ with entries in $\{-1, 0, 1\}$ (not just the typical $\{0, 1\}$). In particular, we set

$$(y_i, \bar{y}_i) = \begin{cases} (y_i, \bar{y}_i) = (0, 1) & \text{if } x_i = 0 \\ (y_i, \bar{y}_i) = (1, -1) & \text{if } x_i = 1. \end{cases} \tag{8}$$

Already this provides a way to perform inference (i.e. to compute arbitrary marginal probabilities in linear time) given a circuit computing $\bar{c}$. We can also construct a new circuit for p by forming the circuit:

$$\bar{c}(x_1, \ldots, x_n, 1 - 2x_1, \ldots, 1 - 2x_n).$$

To see that this is correct, observe

$$\begin{aligned} \bar{c}(x_1, \ldots, x_n, &1 - 2x_1, \ldots, 1 - 2x_n) \\ &= \sum_{T \subseteq [n]} c(x_T) \prod_{i \in T} x_i \prod_{i \notin T} (1 - 2x_i) \\ &= \sum_{T \subseteq S_x} c(x_T) \prod_{i \notin T} (1 - 2x_i) \\ &= \sum_{T \subseteq S_x} c(x_T) (-1)^{\sum_{i \notin T} x_i} \\ &= \sum_{T \subseteq S_x} c(x_T) (-1)^{|S_x| - |T|} \\ &= p(x) \end{aligned}$$

where S_x is the set with characteristic vector $x = (x_1, \ldots, x_n)$. Here the equalities hold for $x \in \{0,1\}^n$ for the following reasons. The first equality is from definition; the second equality holds because for any $T \supset S_x$, there is some $i \in T$ such that $x_i = 0$, and so $\prod_{i \in T} x_i = 0$; the third equality holds because $1 - 2x_i = (-1)^{x_i}$ for $x_i \in \{0, 1\}$; the final equality holds because $T \subseteq S_x$.

We now prove Proposition 3.

Proof. Given a decomposable PC computing F, we construct a decomposable PC for f inductively. For a sum node with $P(x) = \sum_i P_i(x)$, we have

$$\frac{\partial^n}{\partial x_1 \ldots \partial x_n} P(x) = \sum_i \frac{\partial^n}{\partial x_1 \ldots \partial x_n} P_i(x)$$

by linearity of derivatives. For a product node[3] with $P(x) = P_1(x) P_2(x)$, we assume WLOG that x_n is in the scope of P_2

[2]The complexity bound given in Lemma 1 is improved by a factor of n compared to that presented in [Broadrick et al., 2024]; this improvement follows from the use of more efficient homogenization based on polynomial interpolation [Saptharishi, 2015, Lemma 5.4].

[3]We assume product nodes have two children; this can be enforced trivially with minimal effect on circuit size.

(and therefore not P_1, by decomposability). Then for any i we have

$$\begin{aligned}
\frac{\partial}{\partial x_i} P(x) &= \frac{\partial}{\partial x_i} \left[P_1(x) P_2(x) \right] \\
&= P_1(x) \left(\frac{\partial}{\partial x_i} P_2(x) \right) + \left(\frac{\partial}{\partial x_i} P_1(x) \right) P_2(x) \\
&= P_1(x) \frac{\partial}{\partial x_i} P_2(x)
\end{aligned}$$

where the final equality follows as the partial derivative $\left(\frac{\partial}{\partial x_i} P_1(x) \right)$ is zero because x_i is not in the scope of P_1. In such a way, we partition the partial derivatives between the children P_1, P_2. If P is a leaf, we assume the partial derivative can be computed efficiently. Therefore taking a partial derivative of a circuit produces a circuit of the same size, and all n partial derivatives can be taken while maintaining the size of the circuit. $\square$