Causal Inference Using Tractable Circuits

Adnan Darwiche
Computer Science Department
University of California, Los Angeles
darwiche@cs.ucla.edu

Abstract

The aim of this paper is to discuss a recent result which shows that probabilistic inference in the presence of (unknown) causal mechanisms can be tractable for models that have traditionally been viewed as intractable. This result was reported recently in [15] to facilitate model-based supervised learning but it can be interpreted in a causality context as follows. One can compile a non-parametric causal graph into an arithmetic circuit that supports inference in time linear in the circuit size. The circuit is also non-parametric so it can be used to estimate parameters from data and to further reason (in linear time) about the causal graph parametrized by these estimates. Moreover, the circuit size can sometimes be bounded even when the treewidth of the causal graph is not, leading to tractable inference on models that have been deemed intractable previously. This has been enabled by a new technique that can exploit causal mechanisms computationally but without needing to know their identities (the classical setup in causal inference). Our goal is to provide a causality-oriented exposure to these new results and to speculate on how they may potentially contribute to more scalable and versatile causal inference.

1 Introduction

Tractable arithmetic circuits have been receiving an increased attention in AI and computer science more broadly; see [16] for a recent survey. These circuits represent real-valued functions and are called *tractable* because they allow one to answer some hard queries about these functions through linear-time, feed-forward passes on the circuit structure. These circuits were initially compiled from Bayesian networks as proposed in [13, 12] to facilitate probabilistic reasoning. They were later learned from data, starting with [23], and even handcrafted as initially proposed in [30]. Traditional methods for exact probabilistic inference have a complexity which is exponential in *treewidth* (a graph-theoretic parameter that measures the model's connectivity). With the introduction of compiled circuits, one could practically do inference on models whose treewidth can be in the hundreds; see, e.g., [5, 8, 7]. A recent comprehensive, empirical evaluation of at least a dozen probabilistic inference algorithms showed that methods based on circuits are at the forefront in terms of efficiency [1]; see also [18]. What stands behind the efficacy of circuit-based methods is their ability to aggresively exploit the parametric structure of models such as functional dependencies and context-specific independence [4]. This however has limited their utility in contexts where one does not know the model parameters, which is a common case in causal inference. A circuit compilation method was recently proposed in [15] which can exploit a more abstract form of parametric structure: functional dependencies (i.e., causal mechanisms) whose identities are unknown which is the classical setup in causal inference. This new method was able to compile circuits for models whose treewidth is very large without needing to know the model parameters, leading to a major advance on earlier techniques. Our aim in this paper is to provide an intuitive exposure to this new algorithm and its underlying techniques, while placing it in a causality context. Our belief is that a discussion of these new results could lead to a synthesis on how they can further advance causal inference in terms of scalability and versatility. We start first with a review of some core concepts in causality and circuit-based inference and then follow by a discussion of the new results and their significance.

2 Causal Models and Queries

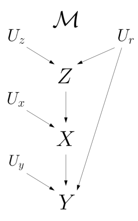
Variables are discrete and denoted by uppercase letters (e.g., X) and their values by lowercase letters (e.g., x). Sets of variables are denoted by boldface, uppercase letters (e.g., $\mathbf{X}$) and their instantiations by boldface, lowercase letters (e.g., $\mathbf{x}$). We will write $x \in \mathbf{x}$ to mean that x is the value of variable X in instantiation $\mathbf{x}$. For a binary variable X, we will use x and $\bar{x}$ to denote X=1 and X=0, respectively.

We next define *Structural Causal Models (SCMs)* following the treatment in [21]; see also [26].

Definition 1. *An SCM is a tuple $\mathcal{M} = (\mathbf{U}, \mathbf{V}, \mathcal{F}, \{Pr(U)\}_{U \in \mathbf{U}})$ where $\mathbf{U}$ and $\mathbf{V}$ are disjoint sets of variables called "exogenous" and "endogenous," respectively; $\mathcal{F}$ contains exactly one function f_V for each endogenous variable V; and $Pr(U)$ are distributions over exogenous variables U. A function f_V is called a "causal mechanism" and it determines the value of variable V based on two sets of inputs, $\mathbf{U}_V \subseteq \mathbf{U}$ and $\mathbf{V}_V \subseteq \mathbf{V}$. That is, the mechanism f_V is a mapping $\mathbf{U}_V, \mathbf{V}_V \mapsto V$.*

The *causal graph* $\mathbb{G}$ of an SCM contains variables $\mathbf{U} \cup \mathbf{V}$ as its nodes. It also contains edges $X \rightarrow V$ for each endogenous variable V and each variable X that is an input to function f_V. We will only deal with SCMs that produce acyclic causal graphs (the inputs of a function cannot depend on its output). Moreover, we will assume that exogenous variables are independent and that only endogenous variables can be observed. A key observation about SCMs is that once we fix the values of exogenous variables, the values of all endogenous variables are also fixed by the causal mechanisms.

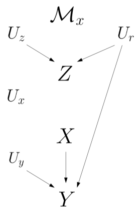
We will next consider an example SCM from [2] where all variables are binary. The endogenous variables are $\mathbf{V} = X, Y, Z$, representing a treatment, the outcome and the presence of hypertension, respectively. The exogenous variables are $\mathbf{U} = U_r, U_x, U_y, U_z$, representing natural resistance to disease (U_r) and sources of variation affecting endogenous variables (U_x, U_y, U_z). The distributions of exogenous variables are $Pr(u_r) = 0.25$, $Pr(u_x) = 0.9$, $Pr(u_y) = 0.7$ and $Pr(u_z) = 0.95$. The three causal mechanisms are given next and they lead to the causal graph on the right:



causal graph

$$\begin{aligned} f_X(Z, U_x) &= z u_x + \bar{z} \bar{u}_x \\ f_Y(X, U_y, U_r) &= x u_r + \bar{x} u_y u_r + \bar{x} \bar{u}_y \bar{u}_r \\ f_Z(U_z, U_r) &= u_z u_r \end{aligned}$$

A key notion in causal inference is the *sub-model* $\mathcal{M}_{\mathbf{z}}$ of an SCM $\mathcal{M}$, where $\mathbf{z}$ is an instantiation of some endogenous variables. This is another SCM obtained from $\mathcal{M}$ by replacing the function for each variable $Z \in \mathbf{Z}$ with the constant function $f_Z = z$, where $z \in \mathbf{z}$. Intuitively, the sub-model $\mathcal{M}_{\mathbf{z}}$ is used to reason about an *intervention* from outside the system modeled by $\mathcal{M}$, which suppresses the causal mechanisms of variables $\mathbf{Z}$ and fixes the values of these variables to $\mathbf{z}$. For example, if we intervene to administer a treatment (x) in the above SCM $\mathcal{M}$, we get a sub-model $\mathcal{M}_x$ in which the mechanism for variable X is replaced with the constant function $f_X = x$. The causal graph of the resulting sub-model is shown on the right.



sub-model

Three types of queries are normally posed on SCMs, which are referred to as *associational, interventional* and *counterfactual* queries. They correspond to what is known as the *causal hierarchy* [26, 28, 27], where each class of queries belongs to a *rung* [28] or *layer* [2] in the hierarchy. We next define the syntax and semantics of these queries, using a slightly different notation than is customary—this will allow us to provide a more uniform and general treatment of these queries.

Definition 2. *An "observational event" has the form $\mathbf{x}$ where $\mathbf{X}$ is a set of endogenous variables. An "interventional event" has the form $\mathbf{y}_{\mathbf{x}}$ where $\mathbf{X}$ and $\mathbf{Y}$ are sets of endogenous variables. A "counterfactual event" has the form $\eta_1, \ldots, \eta_n$ where η_i is an observational or interventional event.*

An observational event $\mathbf{x}$ says that variables $\mathbf{X}$ took the value $\mathbf{x}$. For example, a patient did not take the treatment and died ($\bar{x}, \bar{y}$). An interventional event $\mathbf{y}_{\mathbf{x}}$ says that variables $\mathbf{Y}$ took the value $\mathbf{y}$ after setting variables $\mathbf{X}$ to $\mathbf{x}$ by an intervention. For example, a patient survived after they were given the treatment (y_x). A counterfactual event is a conjunction of events where each could be observational or interventional. For example, a patient who responds to treatment did not take it and died ($y_x, \bar{x}, \bar{y}$).

Definition 3. *A "world" for SCM $\mathcal{M}$ is an instantiation of its exogenous variables. The worlds of an observational event $\mathcal{W}_{\mathcal{M}}(\mathbf{x})$ are those worlds that fix the values of variables $\mathbf{X}$ to $\mathbf{x}$. The worlds of an interventional event $\mathcal{W}_{\mathcal{M}}(\mathbf{y}_{\mathbf{x}})$ are defined as $\mathcal{W}_{\mathcal{M}_{\mathbf{x}}}(\mathbf{y})$. The worlds of a counterfactual event $\mathcal{W}_{\mathcal{M}}(\eta_1, \ldots, \eta_n)$ are defined as $\mathcal{W}_{\mathcal{M}}(\eta_1) \cap \ldots \cap \mathcal{W}_{\mathcal{M}}(\eta_n)$.*

The following table, borrowed from [2], shows all sixteen worlds of the above SCM $\mathcal{M}$, together with their probabilities and the unique states they entail for endogenous variables.

	U_r	U_z	U_x	U_y	Z	X	Y	$P(\mathbf{u})$
1	0	0	0	0	0	1	0	0.001125
2	0	0	0	1	0	1	0	0.002625
3	0	0	1	0	0	0	1	0.010125
4	0	0	1	1	0	0	0	0.023625
5	0	1	0	0	0	1	0	0.021375
6	0	1	0	1	0	1	0	0.049875
7	0	1	1	0	0	0	1	0.192375
8	0	1	1	1	0	0	0	0.448875
9	1	0	0	0	0	1	1	0.000375
10	1	0	0	1	0	1	1	0.000875
11	1	0	1	0	0	0	0	0.003375
12	1	0	1	1	0	0	1	0.007875
13	1	1	0	0	1	0	0	0.007125
14	1	1	0	1	1	0	1	0.016625
15	1	1	1	0	1	1	1	0.064125
16	1	1	1	1	1	1	1	0.149625

Using this table and a similar one for sub-model $\mathcal{M}_x$, we obtain $\mathcal{W}_{\mathcal{M}}(\bar{x}, \bar{y}) = \{\mathbf{u}_4, \mathbf{u}_8, \mathbf{u}_{11}, \mathbf{u}_{13}\}$ and $\mathcal{W}_{\mathcal{M}}(y_x) = \{\mathbf{u}_9, \ldots, \mathbf{u}_{16}\}$ which leads to $\mathcal{W}_{\mathcal{M}}(y_x, \bar{x}, \bar{y}) = \mathcal{W}_{\mathcal{M}}(\bar{x}, \bar{y}) \cap \mathcal{W}_{\mathcal{M}}(y_x) = \{\mathbf{u}_{11}, \mathbf{u}_{13}\}$.

We are now ready to define the probability of any SCM event.

Definition 4. *Let $\mathcal{M}$ be an SCM with exogenous variables $\mathbf{U}$ and distributions $Pr(U)$ for $U \in \mathbf{U}$. The probability of event η with respect to SCM $\mathcal{M}$ is defined as:*

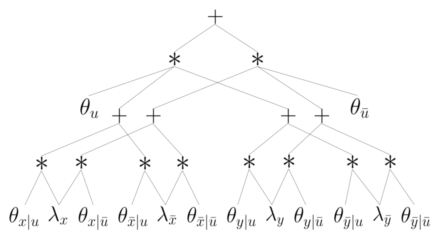
$$Pr(\eta) = \sum_{\mathbf{u} \in \mathcal{W}_{\mathcal{M}}(\eta)} Pr(\mathbf{u}). \tag{1}$$

Hence, $Pr(\bar{x}, \bar{y}) = Pr(\mathbf{u}_4) + Pr(\mathbf{u}_8) + Pr(\mathbf{u}_{11}) + Pr(\mathbf{u}_{13}) = 0.4830$ and $Pr(y_x, \bar{x}, \bar{y}) = 0.0105$. We can now compute the probability that a patient who did not take the treatment and died would have been alive had they been given the treatment, $Pr(y_x|\bar{x}, \bar{y}) = Pr(y_x, \bar{x}, \bar{y})/Pr(\bar{x}, \bar{y}) = 0.0217$. We will focus next on associational and interventional queries, leaving counterfactuals to future work.

3 Causal Inference Using Circuits

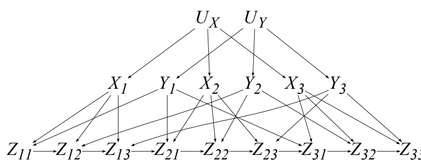
As we just saw, one can answer sophisticated causal queries based on a fully specified SCM. However, such a model may not be available, particularly the identities of causal mechanisms and the distributions over exogenous variables. What is more common is to have the causal graph of an underlying SCM in addition to observational data about endogenous variables. A key task of causal inference is then to draw conclusions based on this limited input, particularly about interventional probabilities which is the task we shall focus on. We will next show how this cross-layer inference can be realized using feed-forward circuits that are compiled from the non-parametric causal graphs of SCMs. The discussion will reveal the significance of compiling the smallest possible circuit for a causal graph. It will also motivate the new compilation algorithm in [15] which is particularly relevant to the causal graphs of SCMs (in contrast to Bayesian networks). We shall discuss and study further this algorithm in Sections 4-6 where we will also contribute to understanding its complexity.

The Circuits of Causal Graphs Consider the causal graph $X \leftarrow U \rightarrow Y$ over binary variables and its compiled arithmetic circuit (AC) shown on the right. This circuit has two types of inputs: *symbolic parameters* θ and *indicators* λ. There are two parameters for exogenous variable U (θ_u and $\theta_{\bar{u}}$) which specify its distribution. There are four parameters for endogenous variable X ($\theta_{x|u}, \theta_{\bar{x}|u}, \theta_{x|\bar{u}}, \theta_{\bar{x}|\bar{u}}$) which specify its causal mechanism. The remaining four parameters specify the mechanism for endogenous variable Y. The indicators correspond to the values of endogenous variables. Variable X has indicators λ_x and $\lambda_{\bar{x}}$ and variable Y has indicators λ_y and $\lambda_{\bar{y}}$. This circuit can compute the probability of any observational event $\eta = \mathbf{x}$ in time linear in the circuit size. We simply set each indicator to 1 if its subscript is compatible with the event η, otherwise we set it to 0, and then evaluate the circuit [13]. For the event $\eta = \bar{x}$, the indicators are set to λ_x=0, $\lambda_{\bar{x}}$=1, λ_y=1, $\lambda_{\bar{y}}$=1. If we (symbolically) evaluate the circuit under this indicator setting, we get $\theta_u \theta_{\bar{x}|u}(\theta_{y|u} + \theta_{\bar{y}|u}) + \theta_{\bar{u}} \theta_{\bar{x}|\bar{u}}(\theta_{y|\bar{u}} + \theta_{\bar{y}|\bar{u}})$ which is the expected result, $Pr(\bar{x})$. The circuit can also compute the probability of any interventional event $\eta = \mathbf{y_x}$ in time linear in the circuit size. This is remarkably simple as well. To compute $Pr(\mathbf{y_x})$, known as the *causal effect,* we first set the


+
*
θ_u
$\theta_{\bar{u}}$
$\theta_{x|u}$ λ_x $\theta_{x|\bar{u}}$ $\theta_{\bar{x}|u}$ $\lambda_{\bar{x}}$ $\theta_{\bar{x}|\bar{u}}$ $\theta_{y|u}$ λ_y $\theta_{y|\bar{u}}$ $\theta_{\bar{y}|u}$ $\lambda_{\bar{y}}$ $\theta_{\bar{y}|\bar{u}}$
arithmetic circuit (AC)

parameters of all variables in $\mathbf{X}$ to 1 and then evaluate the circuit at instantiation $\mathbf{x}, \mathbf{y}$.[1] Suppose that $\mathbf{x} = \bar{x}$ and $\mathbf{y} = \bar{y}$ in our running example. To compute $Pr(\bar{y}_{\bar{x}})$, we evaluate the circuit while setting the parameters for variable X to $\theta_{x|u} = \theta_{\bar{x}|u} = \theta_{x|\bar{u}} = \theta_{\bar{x}|\bar{u}} = 1$, and setting the indicators to λ_x=0, $\lambda_{\bar{x}}$=1, λ_y=0, $\lambda_{\bar{y}}$=1. If we (symbolically) evaluate the circuit under these settings, we get $Pr(\bar{y}_{\bar{x}}) = \theta_u \theta_{\bar{y}|u} + \theta_{\bar{u}} \theta_{\bar{y}|\bar{u}} = Pr(\bar{y})$ which is the expected result (X has no causal effect on Y).

Exploiting Unknown Mechanisms Consider the endogenous variable X in the causal graph we just discussed and its parameters $\theta_{x|u}, \theta_{\bar{x}|u}, \theta_{x|\bar{u}}, \theta_{\bar{x}|\bar{u}}$. Since these parameters specify a causal mechanism, they must all be in $\{0, 1\}$ subject to the constraints $\theta_{x|u} + \theta_{\bar{x}|u} = 1$ and $\theta_{x|\bar{u}} + \theta_{\bar{x}|\bar{u}} = 1$. The main contribution of the new compilation algorithm in [15] is that it can exploit these constraints—without needing to know the specific values of $\theta_{x|u}, \theta_{\bar{x}|u}, \theta_{x|\bar{u}}, \theta_{\bar{x}|\bar{u}}$—to produce circuits whose size can be exponentially smaller compared to methods developed during the last two decades; see, e.g., [13, 5, 6, 11, 32] and [14, Chapters 12,13]. These earlier methods can exploit functional dependencies computationally but only if they know the specific values of parameters. This does not help in a causality context (or a learning context more generally) where we do not know these values. The details of how the algorithm does this are discussed in Sections 4-6. For now, consider the family of causal graphs on the right which generally has variables $U_X, U_Y, X_i, Y_j, Z_{ij}$ for $i, j = 1, \ldots, n$. These models have treewidth $\geq n + 1$ and hence are not accessible to non-parametric inference methods as they take time exponential in n. Moreover, the causal effect of X_1 on Z_{nn} has the only back-door $X_2, \ldots, X_n$. By exploiting (unknown) causal mechanisms, we can now compile these causal graphs into circuits of size $O(n^2)$ (linear in the number of variables), allowing us to compute associational and interventional queries in $O(n^2)$ time. We will say more about this model and back-doors later.



Cross-Layer Inference We next show how to perform cross-layer causal inference using circuits. In a nutshell, we will use the circuit to estimate maximum-likelihood parameters for both exogenous and endogenous variables in the causal graph. We will then plug the estimates into the circuit and use it to compute associational and interventional probabilities in time linear in the circuit size as shown earlier. Suppose $\mathbf{V}$ is the set of observed endogenous variables. Since the causal graph has hidden variables, the maximum-likelihood parameters are not unique so they are not identifiable. However, the distribution $Pr(\mathbf{V})$ is identifiable in this case. That is, even though we may have multiple maximum-likelihood estimates, they all lead to the same distribution $Pr(\mathbf{V})$.[2] This approach will need to be applied carefully though as its validity depends on (1) the specific query of interest, (2) the set of observed variables and (3) the causal graph structure. We will discuss this in detail after elaborating further on the estimation of maximum-likelihood parameters.

Estimation Since each example in a dataset corresponds to an observational event, one can use arithmetic circuits to compute the likelihood function by simply evaluating the circuit at each example (in linear time) and then multiplying the results. The algorithm in [15] facilitates this computation further as it compiles causal graphs into circuits in the form of *tensor graphs.* These are computation graphs in which nodes represent tensor operations instead of arithmetic operations, allowing one to evaluate and differentiate the circuit significantly more efficiently (think of a tensor operation as doing a bulk of arithmetic operations in parallel). Tensor graphs can lead to orders of magnitude speedups in estimation and inference time, especially that they allow batch (parallel) processing of examples—see [15, 9] which compiled circuits with tens of millions of nodes, leading to evaluation times in milliseconds. Backpropagation on arithmetic circuits takes time linear in the circuit size. Moreover, the partial derivatives with respect to circuit parameters correspond to marginals over families in the causal graph (nodes and their parents) [13], which is all that one needs to compute parameter updates for the EM algorithm; see, for example, [14, Eq. 17.7]. Hence, one can use methods such as gradient descent and EM to seek maximum-likelihood estimates, but the efficacy of these methods needs further investigation under the stated conditions including finite data.

[1]This method follows directly from the mutilation semantics of interventions [26, Section 1.3.1] and the polynomial semantics of arithmetic circuits [13, 10]. It is a slight variation on [31] which also emulates interventions by adjusting model parameters; see also [37].

[2]Ying Nian Wu provided the following argument for infinite data. Let $Pr_D(\mathbf{V})$ be the data distribution and $Pr_\theta(\mathbf{U}, \mathbf{V})$ be the model so $Pr_\theta(\mathbf{V}) = \sum_{\mathbf{U}} Pr_\theta(\mathbf{U}, \mathbf{V})$. Maximum-likelihood estimation is equivalent to minimizing the KL divergence $KL(Pr_D(\mathbf{V})|Pr_\theta(\mathbf{V}))$. If θ is not identifiable, then all solutions of θ belong to an equivalence class that minimizes the KL divergence and they all give the same marginal $Pr_\theta(\mathbf{V})$.

Identifiability A central question in causal inference is whether interventional probabilities can be identified based on a causal graph and (infinite) observational data on the endogenous variables $\mathbf{V}$. Intuitively, identifiability means that interventional probabilities can be computed using any parameterization of the causal graph (or circuit) that yields the true marginal distribution $Pr(\mathbf{V})$; see [26, Def 3.2.4] for a formal definition. A well behaved case arises when each exogenous variable feeds into at most one causal mechanism. These models are said to be *Markovian* and the case is termed *no unobserved confounders.* Interventional probabilities are always identifiable for Markovian models, so we can always compute the causal effect $Pr(\mathbf{y}_\mathbf{x})$ for these models using circuits parameterized by maximum-likelihood parameters.[3] If an exogenous variable feeds into more than one causal mechanism, the model is said to be *semi-Markovian.* In this case, interventional probabilities are identifiable only when certain conditions are met. The causal graph on the right corresponds to a semi-Markovian model since U feeds into the causal mechanisms for both X and Z. The causal effect $Pr(y_x)$ is identifiable since $Pr(y_x) = \sum_z Pr(y|x,z)Pr(z)$. However, the causal effect $Pr(x_z)$ is not identifiable as it cannot be uniquely determined based on the causal graph and the distribution $Pr(X, Y, Z)$. The *do-calculus* provides a complete and efficient characterization of identifiable, interventional probabilities based on observational data [25, 35, 34]; see also [20].[4] It is based on a set of rules that can be used to transform an interventional probability into a formula that includes only associational probabilities (we will call this an *identifiability formula*). If the rules fail to make such a derivation, then the interventional probability is not identifiable. Other methods such as *back-door* [24] and *front-door* [29] provide simpler but incomplete tests and lead to simple identifiability formulas. For example, the back-door and front-door formulas have the forms $Pr(\mathbf{y}_\mathbf{x}) = \sum_\mathbf{z} Pr(\mathbf{y}|\mathbf{x},\mathbf{z})Pr(\mathbf{z})$ and $Pr(\mathbf{y}_\mathbf{x}) = \sum_\mathbf{z} Pr(\mathbf{z}|\mathbf{x}) \sum_{\mathbf{x}'} Pr(\mathbf{y}|\mathbf{x}',\mathbf{z})Pr(\mathbf{x}')$, where $\mathbf{Z}$ is called a back-door or front-door, respectively. A more refined identifiability test that utilizes context-specific independence relations was proposed recently [36], thus expanding the reach of cross-layer causal inference. These relations correspond to equating certain parameters and can be integrated into circuits [5] to reduce the number of estimands. In summary, we can compute causal effects on semi-Markovian models using circuits parameterized by maximum-likelihood estimates, but only for identifiable queries as licensed by the do-calculus or a more refined identifiability procedure.

$$\begin{tikzcd}
U \arrow[rr] \arrow[dr] & & Z \arrow[dl] \arrow[dr] & \\
& X \arrow[rr] & & Y
\end{tikzcd}$$

Identifiability Formulas vs Circuits Most identifiability procedures yield formulas (also called the *effect estimand*) which play three roles: they provide a proof of identifiability; they point to endogenous variables whose measurement guarantees identifiability; and they allow one to evaluate the causal effect by estimating quantities that populate such formulas. Back-door and front-door formulas have simple forms (albeit exponential sums) but the do-calculus and the procedure in [36] may generate identifiability formulas that are much more complex. Some procedures do not even aim to produce identifiability formulas; e.g., [19]. To evaluate causal effects using circuits, one only needs an identifiability test not a formula. That is, one estimates circuit parameters only once and then uses the parametrized circuit to answer any identifiable query in time linear in the circuit size—regardless of how complex the identifiability formula may be and without needing to have access to one.[5] Additional knowledge such as context-specific independence and known mechanisms can be directly integrated into the circuit [5], which can only improve the quality of estimates under finite data. At its core, this use of circuits amounts to computing causal effects based on the classical method of mutilating causal graphs, armed by an observation and an advance. The observation is that using maximum-likelihood parameters is sound if the causal effect is identifiable. The advance is that we can now perform this computation much more efficiently due to exploiting unknown mechanisms.

The Circuit Compilation Process We will next discuss the circuit compilation algorithm introduced recently in [15]. We will focus on the key insights behind the algorithm and slightly adjust it to suit our current objectives (the original algorithm targetted specific queries as is typically demanded in a supervised learning setting). In a nutshell, the algorithm is based on the classical algorithm of *variable elimination* (VE) with two exceptions. First, we will use VE symbolically by working with symbolic parameters instead of numeric ones. Second, we will empower VE by two new theorems based on unknown causal mechanisms which can reduce its complexity exponentially. We will review VE in Section 4 and then present the new theorems and compilation algorithm in Sections 5 and 6.

[3]See [34, Corollary 4] for a weaker, necessary condition that guarantees identifiability of all causal effects.

[4]See also [22] for a treatment of identifiability based on both observational and interventional data and [3] for a survey that considers other tasks such as the evaluation of soft interventions.

[5]Using circuits in this manner requires fixing the cardinality of variables including exogenous ones.

4 The Variable Elimination Algorithm (VE)

VE operates on causal graphs which are parameterized by factors. A *factor* over variables $\mathbf{X}$ is a function $f(\mathbf{X})$ that maps each instantiation $\mathbf{x}$ into a number $f(\mathbf{x})$. For each node X and its parents $\mathbf{P}$ in the causal graph, we need a factor $f_X(X, \mathbf{P})$ where $f_X(x, \mathbf{p}) = Pr(x|\mathbf{p})$. For example, factor $f(U)$ on the right specifies the distribution $Pr(U)$ for exogenous variable U and factor $g(XY)$ specifies the mechanism for endogenous variable Y: $x_0 \mapsto y_1$, $x_1 \mapsto y_0$. VE is based two factor operations: multiplication and sum-out. The *product* of factors $f(\mathbf{X})$ and $g(\mathbf{Y})$ is another factor $h(\mathbf{Z})$, where $\mathbf{Z} = \mathbf{X} \cup \mathbf{Y}$ and $h(\mathbf{z}) = f(\mathbf{x})g(\mathbf{y})$ for the unique instantiations $\mathbf{x}$ and $\mathbf{y}$ that are compatible with instantiation $\mathbf{z}$. *Summing-out* variables $\mathbf{Y} \subseteq \mathbf{X}$ from factor $f(\mathbf{X})$ yields another factor $g(\mathbf{Z})$, where $\mathbf{Z} = \mathbf{X} \setminus \mathbf{Y}$ and $g(\mathbf{z}) = \sum_{\mathbf{y}} f(\mathbf{yz})$. We use $\sum_{\mathbf{Y}} f$ to denote the resulting factor g.

U	$f(U)$	
u_0	0.3	θ_{u_0}
u_1	0.1	θ_{u_1}
u_2	0.6	θ_{u_2}

X	Y	$g(XY)$	
x_0	y_0	0	$\theta_{y_0\|x_0}$
x_0	y_1	1	$\theta_{y_1\|x_0}$
x_1	y_0	1	$\theta_{y_0\|x_1}$
x_1	y_1	0	$\theta_{y_1\|x_1}$

The *joint distribution* of a parametrized causal graph is simply the product of its factors. The causal graph in Figure 1(b) has factors $f_A(A)$, $f_B(AB)$, $f_C(AC)$, $f_D(BCD)$ and $f_E(CE)$. Its joint distribution is $Pr(ABCDE) = f_A f_B f_C f_D f_E$. To record an observation $X{=}x$, we use an auxiliary *evidence factor* $\lambda_X(X)$ with $\lambda_X(x) = 1$ and $\lambda_X(x') = 0$ for $x' \neq x$. A *posterior distribution* is obtained by normalizing the product of all factors in the causal graph including evidence factors. Suppose we have evidence $\mathbf{e}$ on variables A and E in Figure 1(b). The posterior $Pr(D|\mathbf{e})$ is obtained by evaluating then normalizing the expression $\sum_{ABCE} \lambda_A \lambda_E f_A f_B f_C f_D f_E$. VE tries to evaluate such expressions efficiently [38, 17] based on two theorems; see, e.g., [14, Chapter 6].

The first theorem allows us to sum out variables in any order. The second theorem allows us to pull out factors from sums, which can lead to exponential savings in time and space.

Theorem 1. $\sum_{\mathbf{XY}} f = \sum_{\mathbf{X}} \sum_{\mathbf{Y}} f = \sum_{\mathbf{Y}} \sum_{\mathbf{X}} f$.

Theorem 2. *If variables $\mathbf{X}$ appear in factor f but not in factor g, then $\sum_{\mathbf{X}} f \cdot g = g \sum_{\mathbf{X}} f$.*

Consider the expression $\sum_{ABDE} f(ACE)f(BCD)$. A direct evaluation multiplies the two factors to yield $f(ABCDE)$ then sums out variables $ABDE$. Using Theorem 1, we can arrange the expression into $\sum_{AE} \sum_{BD} f(ACE)f(BCD)$. Using Theorem 2, we can arrange it further into $\sum_{AE} f(ACE) \sum_{BD} f(BCD)$ which is more efficient to evaluate. If we eliminate all variables using order π, and if the largest factor constructed in the process has $w + 1$ variables, then w is called the *width* of order π. The smallest width attained by any elimination order corresponds to the *treewidth* of the causal graph. The best time complexity that can be attained by VE is $O(n \exp(w))$, where n is the number of variables and w is the causal graph treewidth. This holds for any other non-parametric method known today (i.e., inference methods that do not exploit the graph parameters).

5 Variable Elimination with Causal Mechanisms

We next present two recent results that allow us to simplify expressions beyond what is permitted by Theorems 1 and 2, leading to a tighter complexity based on what we shall call the *causal treewidth.* We will use $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$ to denote sets of factors, where each set is interpreted as a product of its factors.

Definition 5. *A factor $f(X, \mathbf{P})$ is said to be a "mechanism" for variable X iff all numbers in the factor are in $\{0, 1\}$ and $\sum_x f(x, \mathbf{p}) = 1$ for every instantiation $\mathbf{p}$.*

Theorem 3 ([15])**.** *Let f be a mechanism for variable X. If $f \in \mathcal{G}$ and $f \in \mathcal{H}$, then $\mathcal{G} \cdot \mathcal{H} = \mathcal{G} \sum_X \mathcal{H}$.*

According to this result, if a mechanism for X appears in both parts of a product, then variable X can be summed out from one part without changing the value of the product. This has a key corollary.

Corollary 1. *Let f be a mechanism for X. If $f \in \mathcal{G}$ and $f \in \mathcal{H}$, then $\sum_X \mathcal{G} \cdot \mathcal{H} = (\sum_X \mathcal{G})(\sum_X \mathcal{H})$.*

That is, if a mechanism for X appears in both parts of a product, we can sum out variable X from the product by independently summing it out from each part. This is a remarkable addition to the algorithm of variable elimination which has been under study for a few decades now. Corollary 1 may appear unusable as it is predicated on multiple occurrences of a mechanism whereas the factors of a causal graph contain a single mechanism for each endogenous variable. This is where the second result comes in: *replicating* mechanisms in a product does not change the product value.

Theorem 4 ([15])**.** *For mechanism f, if $f \in \mathcal{G}$, then $f \cdot \mathcal{G} = \mathcal{G}$.*

For an example that uses these theorems, consider the expression $\alpha = \sum_X f(XY)g(XZ)h(XW)$. VE has to multiply all three factors before summing out variable X, leading to a factor over four variables $XYZW$. However, if factor f is a mechanism for variable X, then we can replicate it by Theorem 4: $\alpha = f(XY)g(XZ)f(XY)h(XW)$. Corollary 1 then gives $\alpha = \sum_X f(XY)g(XZ) \sum_X f(XY)h(XW)$. Hence, we can now evaluate expression α without having to construct any factor over more than three variables. This technique can more generally lead to exponential savings since the size of a factor is exponential in the number of its variables.

We will refer to the extension of VE with Theorems 3 and 4 as VEC (**V**ariable **E**limination for **C**ausality). We emphasize that these new theorems do not require the values of parameters (i.e., specific mechanisms). They only need to know if a variables is functionally determined by its parents.

6 Compiling Causal Graphs Into Circuits

We next show how VE/VEC can be used *symbolically* to compile non-parametric causal graphs into arithmetic circuits with symbolic parameters. We will first show this concretely on a small example using VE, then discuss a general compilation algorithm based on VE and finally based on VEC.

Consider the causal graph $U \rightarrow V$ with binary variables. The parameters of this graph are given by the factors $f(U)$ and $g(UV)$ shown on the right. We also added an evidence factor for endogenous variable V as it will be measured. These factors have symbolic parameters instead of numeric ones.

U	$f(U)$
u	θ_u
$\bar{u}$	$\theta_{\bar{u}}$

U	V	$g(UV)$
u	v	$\theta_{v\|u}$
u	$\bar{v}$	$\theta_{\bar{v}\|u}$
$\bar{u}$	v	$\theta_{v\|\bar{u}}$
$\bar{u}$	$\bar{v}$	$\theta_{\bar{v}\|\bar{u}}$

V	$h(V)$
v	λ_v
$\bar{v}$	$\lambda_{\bar{v}}$

We will further overload the $+$ and $*$ operators so they now construct circuit nodes instead of performing numeric operations. That is, each entry in the above factors can be viewed as a leaf circuit node. When multiplying, say, node θ_u with node $\theta_{v|u}$, we construct a circuit node with $*$ as its label and nodes θ_u, $\theta_{v|u}$ as its children. And similarly for addition. We can now get a circuit for the causal graph by multiplying all its factors, including evidence factors, and then summing out all variables, $AC = \sum_{UV} f(U)g(UV)h(V)$. The resulting factor AC will have a single entry which contains the root of compiled circuit $\lambda_v * \theta_u * \theta_{v|u} + \lambda_{\bar{v}} * \theta_u * \theta_{\bar{v}|u} + \lambda_v * \theta_{\bar{u}} * \theta_{v|\bar{u}} + \lambda_{\bar{v}} * \theta_{\bar{u}} * \theta_{\bar{v}|\bar{u}}$. The equivalent expression $AC = \sum_V h(V) \sum_U f(U)g(UV)$ gives the circuit $\lambda_v * (\theta_u * \theta_{v|u} + \theta_{\bar{u}} * \theta_{v|\bar{u}}) + \lambda_{\bar{v}} * (\theta_u * \theta_{\bar{v}|u} + \theta_{\bar{u}} * \theta_{\bar{v}|\bar{u}})$. Hence, the size and shape of a compiled circuit depend on how we schedule factor operations (multiplication and sum-out). The symbolic use of VE to compile circuits was initially proposed in [6]. This method was recently refined in [15] by (1) *scheduling* factor operations based on a specific class of *binary jointrees* [33] and (2) allowing one to compile circuits using VEC by *thinning* the jointree. We will explain this advance after a brief review of (binary) jointrees.

Jointrees A binary jointree is a tree in which each node is either a *leaf* (has a single neighbor) or *internal* (has three neighbors). We will require the leaf nodes to be in one-to-one correspondence with the factors of a causal graph, including replicated factors but excluding evidence factors. As we show next, the topology of a binary jointree determines all its properties, including the set of variables attached to each node, called a *cluster,* and the set of variables attached to each edge, called a *separator.* Figure 1(a) depicts a binary jointree for the causal graph in Figure 1(b). Following the convention in [15], this jointree is layed out so that each internal node has two neighbors below it (children) and the third neighbor above it (parent). The leaf nodes of this jointree are numbered $1, 6, 8, 9, 10, 11, 12$ and correspond to the causal graph factors, $f_B, f_A, f_C, f_E, f_C, f_B, f_D$ (we replicated the factors for B and C). The separator for edge (i, j) between node i and its parent j is denoted $\texttt{sep}(i)$ and contains variables that are shared between factors on both sides of the edge (i, j). For example, $\texttt{sep}(3) = \{A, C\}$ as these are the variables shared between factors at leaves $\{6, 9, 10\}$ and factors at leaves $\{1, 8, 11, 12\}$. The cluster of node i is denoted $\texttt{cls}(i)$. The cluster of a leaf node is the variables of its associated factor. The cluster of an internal node is the union of separators connected to its children. For example, for leaf node 9 with factor f_E, $\texttt{cls}(9) = \texttt{vars}(f_E) = \{C, E\}$. Moreover, for internal node 7 with children 11 and 12, $\texttt{cls}(7) = \texttt{sep}(11) \cup \texttt{sep}(12) = \{A, B, C\}$.

Scheduling Given a binary jointree, VE (and later VEC) schedules its operations as follows. Visiting nodes bottom-up in the jointree, each node i computes a factor $f(i)$ and sends it to its parent. A leaf node i computes $f(i)$ by projecting its associated factor on $\texttt{sep}(i)$. For example, $f(9) = \sum_E f_E \lambda_E$. An internal node i computes $f(i)$ by multiplying the factors it receives from its children and then projecting the product on $\texttt{sep}(i)$. For example, $f(2) = \sum_C f(3)f(4)$. This process terminates at the

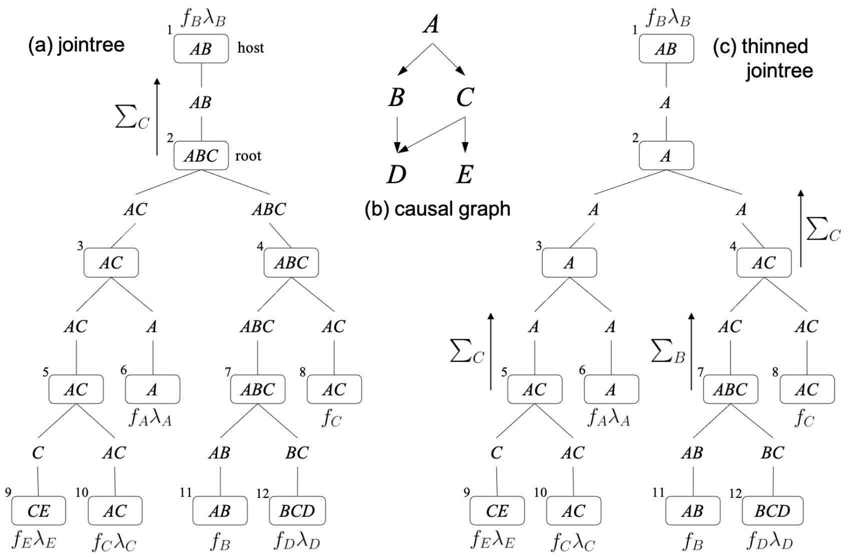


Figure 1: A causal graph (middle) with a jointree (left) and a thinned jointree (right). The mechanisms for variables B and C, f_B and f_C, are replicated twice in the jointrees.

top leaf node r which multiplies the factor it receives from its single child c with its own factor and then projects the product on the empty set. In Figure 1(a), the final computation is $\sum_{AB} f_B \lambda_B f(2)$. Denoting the factor at leaf node i by $\mathcal{F}_i$, this process yields the factor $AC = \sum_{\texttt{cls}(r)} \mathcal{F}_r f(c)$, where

$$f(i) = \sum_{\texttt{cls}(i)\setminus\texttt{sep}(i)} \mathcal{F}_i \quad \text{if } i \text{ is leaf;} \quad f(i) = \sum_{\texttt{cls}(i)\setminus\texttt{sep}(i)} f(c_1)f(c_2) \quad \text{if } i \text{ has children } c_1, c_2.$$

The final factor AC has a single entry which contains the root of the compiled circuit as shown earlier. Moreover, the size of this circuit is determined by the jointree clusters and separators. Each cluster/separator contributes a number of multiplication/addition nodes that is exponential in the cluster/separator size. In a jointree, the largest cluster dominates the largest separator and the size of the largest cluster minus 1 is called the jointree *width.* Furthermore, the smallest width attained by any jointree corresponds to the treewidth of the causal graph; see, [14, Chapter 9]. Hence, the complexity of this compilation method is exponential in the treewidth of the causal graph.

Thinning This complexity was recently significantly improved by exploiting (unknown) causal mechanisms [15]. The basic idea is to *thin* the jointree by shrinking its separators (and hence clusters) while maintaining the correctness of compiled circuit. The thinning process is based on Theorems 3 and 4 and can lead to an exponential reduction in the circuit size. To see the key insight behind this thinning process, consider node 2 in the jointree of Figure 1(a). The separators $\texttt{sep}(3)$ and $\texttt{sep}(4)$ of its children both contain variable C. Hence, the factors $f(3)$ and $f(4)$ sent by these children to node 2 both contain variable C. Since C does not appear in $\texttt{sep}(2)$ it gets summed out at node 2 so it does not appear in the factor $f(2)$ that this node sends to its parent. However, since we have two replicas of the mechanism for variable C at leaf nodes 8 and 10 (as licensed by Theorem 4), we can sum out C earlier, at nodes 4 and 5 (as licensed by Theorem 3). This means that C can be removed from $\texttt{sep}(4)$, $\texttt{sep}(5)$ and also $\texttt{sep}(3)$. We can similarly sum out variable B at node 7, which removes it from $\texttt{sep}(7)$, $\texttt{sep}(4)$ and $\texttt{sep}(2)$. The shrinking of separators causes clusters to shrink as well, leading to the thinned jointree in Figure 1(c) and a corresponding smaller circuit compilation.

Causal Treewidth The attained reduction in complexity depends on (1) the number of replicas for each mechanism; (2) the used binary jointree; and (3) how the jointree is thinned. Corresponding heuristics were proposed in [15] and the resulting algorithm was shown to yield exponential reductions in the size of compiled circuits on a number of benchmarks (elimination orders and jointrees are also constructed using heuristics since finding optimal ones is NP-hard). This motivates a new measure of complexity which we call the *causal treewidth.* We define this as the smallest width attained by any thinned jointree for a given causal graph. We next complement the empirical findings in [15] by showing that causal treewidth dominates treewidth and can be bounded when the treewidth is not.

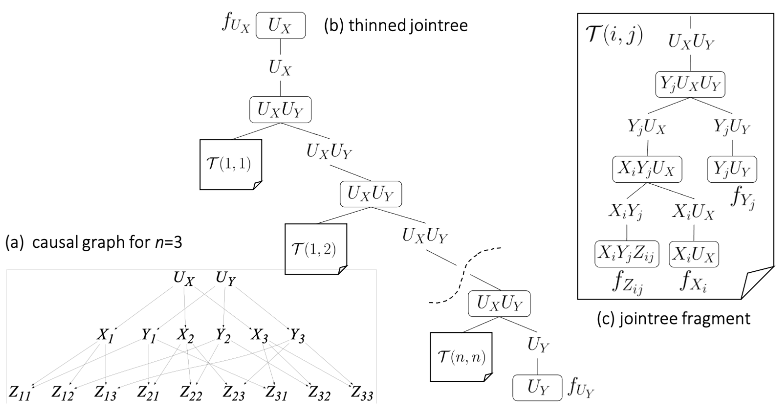


Figure 2: A causal graph and its thinned jointree. Mechanisms f_{X_i} and f_{Y_j} are replicated n times.

Theorem 5. *The causal treewidth is no greater than treewidth. Moreover, there is a family of causal graphs with $n^2 + 2n + 1$ variables, treewidth $n + 1$ and causal treewidth 2 where n is an integer ≥ 1.*

Proof Sketch. Consider a causal graph $\mathbb{G}$ with treewidth w. Without replicating mechanisms, we can always get a (thinned) jointree with width w. Hence, the causal treewidth of $\mathbb{G}$ is $\leq w$. To show the second part of the theorem, consider the family of causal graphs $\mathbb{G}_n$ with exogenous variables U_X, U_Y and endogenous variables X_i, Y_j, Z_{ij} for $i, j = 1, \ldots, n$ ($2 + 2n + n^2$ variables), and edges $U_X \to X_i$, $U_Y \to Y_j$, $X_i \to Z_{ij}$, $Y_j \to Z_{ij}$. Figure 2(a) depicts $\mathbb{G}_3$. We next show that $\mathbb{G}_n$ has treewidth $n+1$ based on standard techniques for treewidth; see, e.g., [14, Chapter 9]. The moral graph of $\mathbb{G}_n$ is obtained by dropping edge directions and connecting every pair of nodes X_i and Y_j by an undirected edge. Each Z_{ij} has only two (connected) neighbors in the moral graph (X_i and Y_j) so it is a simplicial node. Hence, there must exist an optimal elimination order that starts with nodes Z_{ij} [14, Section 9.3.2]. After eliminating all Z_{ij}, nodes U_X and U_Y will each have n neighbors, and nodes X_i and Y_j will each have $n + 1$ neighbors. A simple argument shows that eliminating these variables in any order from the moral graph will create a clique over $n+1$ variables so the treewidth is $\geq n+1$. One can easily verify that the elimination order $Z_{11}, \ldots, Z_{nn}, U_X, Y_1, \ldots, Y_n, U_Y, X_1, \ldots, X_n$ has width $n + 1$ so the treewidth of $\mathbb{G}_n$ is $n + 1$. Figure 2(b) depicts a thinned jointree for $\mathbb{G}_n$ with width 2, which results from cascading n^2 instances of the jointree fragment $\mathcal{T}(i, j)$ in Figure 2(c). This fragment contains the mechanism for Z_{ij} and replicas of the mechanisms for X_i and Y_j. This thinned jointree is optimal since the mechanism for Z_{ij} contains 3 variables so any thinned jointree must have a cluster of size ≥ 3. Hence, the causal treewidth of $\mathbb{G}_n$ is 2. □

Theorem 5 effectively says that circuits compiled by VEC are no larger than those compiled by VE and can be exponentially smaller. We finally note that a variation $\mathbb{G}'_n$ on $\mathbb{G}_n$ was shown in Section 3 with additional edges between variables Z_{ij}. The treewidth of $\mathbb{G}'_n$ must be $\geq n + 1$ yet has a thinned jointree of width 4 (constructed by the algorithm in [15]) so its causal treewidth is ≤ 4. Recall that for this family of causal graphs $\mathbb{G}'_n$, the causal effect of X_1 on Z_{nn} has the only back-door $X_2, \ldots, X_n$ so it has a back-door formula with a sum that is exponential in n and a circuit of size $O(n^2)$.

7 Conclusion

We discussed recent techniques that can exploit causal mechanisms computationally without having to know their identities, which is the classical setup in causal inference. We also showed how one can use these techniques to compile non-parametric causal graphs into circuits that can be used to estimate parameters from data and to perform cross-layer causal inference in time linear in the circuit size. Our aim was to provide an intuitive exposure to these techniques to a causality audience who may not be as familiar with them, with the hope that this may lead to a synthesis on how tractable arithmetic circuits can aid causal inference in reaching higher levels of scalability and versatility.

Acknowledgements I wish to thank Elias Bareinboim, Yizuo Chen, Scott Mueller, Judea Pearl and Jin Tian for useful discussions and feedback. This work has been partially supported by NSF grant #ISS-1910317 and ONR grant #N00014-18-1-2561.

References

[1] Durgesh Agrawal, Yash Pote, and Kuldeep S. Meel. Partition function estimation: A quantitative study. In *IJCAI*, pages 4276–4285. ijcai.org, 2021.

[2] E. Bareinboim, Juan David Correa, D. Ibeling, and Thomas F. Icard. On Pearl's hierarchy and the foundations of causal inference. 2021. Technical Report, R-60, Colombia University.

[3] Elias Bareinboim and Judea Pearl. Causal inference and the data-fusion problem. *Proc. Natl. Acad. Sci. USA*, 113(27):7345–7352, 2016.

[4] Craig Boutilier, Nir Friedman, Moisés Goldszmidt, and Daphne Koller. Context-specific independence in Bayesian networks. *CoRR*, abs/1302.3562, 2013.

[5] Mark Chavira and Adnan Darwiche. Compiling Bayesian networks with local structure. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 1306–1312. Professional Book Center, 2005.

[6] Mark Chavira and Adnan Darwiche. Compiling Bayesian networks using variable elimination. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2443–2449, 2007.

[7] Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artif. Intell.*, 172(6-7):772–799, 2008.

[8] Mark Chavira, Adnan Darwiche, and Manfred Jaeger. Compiling relational Bayesian networks for exact inference. *Int. J. Approx. Reason.*, 42(1-2):4–20, 2006.

[9] Yizuo Chen, Arthur Choi, and Adnan Darwiche. Supervised learning with background knowledge. In *PGM*, 2020.

[10] Arthur Choi and Adnan Darwiche. On relaxing determinism in arithmetic circuits. In *Proceedings of the Thirty-Fourth International Conference on Machine Learning (ICML)*, pages 825–833, 2017.

[11] Arthur Choi, Doga Kisa, and Adnan Darwiche. Compiling probabilistic graphical models using sentential decision diagrams. In *ECSQARU*, volume 7958 of *Lecture Notes in Computer Science*, pages 121–132. Springer, 2013.

[12] Adnan Darwiche. A logical approach to factoring belief networks. In Dieter Fensel, Fausto Giunchiglia, Deborah L. McGuinness, and Mary-Anne Williams, editors, *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25, 2002*, pages 409–420. Morgan Kaufmann, 2002.

[13] Adnan Darwiche. A differential approach to inference in Bayesian networks. *J. ACM*, 50(3):280–305, 2003.

[14] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

[15] Adnan Darwiche. An advance on variable elimination with applications to tensor-based computation. In *ECAI*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2559–2568. IOS Press, 2020.

[16] Adnan Darwiche. Tractable Boolean and arithmetic circuits. In Pascal Hitzler and Md Kamruzzaman Sarker, editors, *Neuro-symbolic Artificial Intelligence: The State of the Art*. Frontiers in Artificial Intelligence and Applications. IOS Press, 2022. In print.

[17] Rina Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 211–219, 1996.

[18] Paulius Dilkas and Vaishak Belle. Weighted model counting with conditional weights for Bayesian networks. In *UAI*, 2021.

[19] Joseph Y. Halpern. Axiomatizing causal reasoning. *J. Artif. Intell. Res.*, 12:317–337, 2000.

[20] Yimin Huang and Marco Valtorta. Identifiability in causal bayesian networks: A sound and complete algorithm. In *AAAI*, pages 1149–1154. AAAI Press, 2006.

[21] Madelyn Glymour Judea Pearl and Nicholas P. Jewell. *Causal Inference in Statistics: A Primer*. Wiley, 2016.

[22] Sanghack Lee, Juan D. Correa, and Elias Bareinboim. General identifiability with arbitrary surrogate experiments. In *UAI*, volume 115 of *Proceedings of Machine Learning Research*, pages 389–398. AUAI Press, 2019.

[23] Daniel Lowd and Pedro M. Domingos. Learning arithmetic circuits. In *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 383–392, 2008.

[24] Judea Pearl. [bayesian analysis in expert systems]: Comment: Graphical models, causality and intervention. *Statistical Science*, 8(3):266–269, 1993.

[25] Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995.

[26] Judea Pearl. *Causality*. Cambridge University Press, 2000.

[27] Judea Pearl. The seven tools of causal inference, with reflections on machine learning. *Commun. ACM*, 62(3):54–60, 2019.

[28] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, 2018.

[29] Judea Pearl and James M. Robins. Probabilistic evaluation of sequential plans from causal models with hidden variables. In Philippe Besnard and Steve Hanks, editors, *UAI '95: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, Montreal, Quebec, Canada, August 18-20, 1995*, pages 444–453. Morgan Kaufmann, 1995.

[30] Hoifung Poon and Pedro M. Domingos. Sum-product networks: A new deep architecture. In *UAI*, pages 337–346. AUAI Press, 2011.

[31] Biao Qin. Differential semantics of intervention in Bayesian networks. In *IJCAI*, pages 710–716. AAAI Press, 2015.

[32] Yujia Shen, Arthur Choi, and Adnan Darwiche. Tractable operations for arithmetic circuits of probabilistic models. In *NIPS*, pages 3936–3944, 2016.

[33] Prakash P. Shenoy. Binary join trees. In *UAI*, pages 492–499. Morgan Kaufmann, 1996.

[34] Ilya Shpitser and Judea Pearl. Identification of joint interventional distributions in recursive semi-markovian causal models. In *AAAI*, pages 1219–1226. AAAI Press, 2006.

[35] Jin Tian and Judea Pearl. A general identification condition for causal effects. In *AAAI/IAAI*, pages 567–573. AAAI Press / The MIT Press, 2002.

[36] Santtu Tikka, Antti Hyttinen, and Juha Karvanen. Identifying causal effects via context-specific independence relations. In *NeurIPS*, pages 2800–2810, 2019.

[37] Benjie Wang, Clare Lyle, and Marta Kwiatkowska. Provable guarantees on the robustness of decision rules to causal interventions. In *IJCAI*, pages 4258–4265. ijcai.org, 2021.

[38] Nevin Lianwen Zhang and David Poole. Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, 1996.