Journal of Machine Learning Research 18 (2017) 1-30 Submitted 4/16; Revised 3/17; Published 4/17

Simplifying Probabilistic Expressions in Causal Inference

Santtu Tikka SANTTU.TIKKA@JYU.FI
Juha Karvanen JUHA.T.KARVANEN@JYU.FI
Department of Mathematics and Statistics
P.O.Box 35 (MaD) FI-40014 University of Jyvaskyla, Finland

Editor: Peter Spirtes

Abstract

Obtaining a non-parametric expression for an interventional distribution is one of the most fundamental tasks in causal inference. Such an expression can be obtained for an identifiable causal effect by an algorithm or by manual application of do-calculus. Often we are left with a complicated expression which can lead to biased or inefficient estimates when missing data or measurement errors are involved.

We present an automatic simplification algorithm that seeks to eliminate symbolically unnecessary variables from these expressions by taking advantage of the structure of the underlying graphical model. Our method is applicable to all causal effect formulas and is readily available in the R package causaleffect.

Keywords: simplification, probabilistic expression, causal inference, graphical model, graph theory

A modification of (Tikka and Karvanen, 2017b) published in the Journal of Machine Learning Research.

1. Introduction

Symbolic derivations resulting in complicated expressions are often encountered in many fields working with mathematical notation. These expressions can be derived manually or they can be outputs from a computer algorithm. In both cases, the expressions may be correct but unnecessarily complex in a sense that some unrecognized identities or properties would lead to simpler expressions.

We will consider simplification in the context of causal inference in graphical models (Pearl, 2009). Advances in causal inference have led to algorithmic solutions to problems such as identifiability of causal effects and conditional causal effects (Huang and Valtorta, 2006; Shpitser and Pearl, 2006a,b), z-identifiability (Bareinboim and Pearl, 2012), transportability and meta-transportability (Bareinboim and Pearl, 2013b,a) among others. The aforementioned algorithmic solutions operate symbolically on the joint distribution of the variables of interest and return expressions for the desired queries. These algorithms have been previously implemented in the R package causaleffect (Tikka and Karvanen, 2017a). Another implementation of an identifiability algorithm can be found in the CIBN software by Jin Tian and

©2017 Santtu Tikka and Juha Karvanen.

License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/. Attribution requirements are provided at http://jmlr.org/papers/v18/16-166.html.

Lexin Liu freely available from `http://web.cs.iastate.edu/~jtian/Software/CIBN.htm`. However, the algorithms themselves are imperfect in a sense that they often output an expression that is complicated and far from ideal. The question is whether there exists a simpler expression that is still a solution to the original problem.

Simplification of expressions may provide significant benefits. First, a simpler expression can be understood and reported more easily. Second, evaluating a simpler expression will be less of a computational burden due to reduced dimensionality of the problem. Third, in situations where estimation of causal effects is of interest and missing data is a concern, eliminating variables with missing data from the expression has clear advantages. The same applies to variables with measurement error.

We begin with presenting in Section 2 a general form of probabilistic expressions that are often encountered in causal inference. In this paper probabilistic expressions are formed by products of non-parametric conditional distributions of some variables and summations over the possible values of these variables. Simplification in this case is the process of eliminating terms from these expressions by carrying out summations. As our expressions correspond to causal effects, the expressions themselves take a specific form.

Causal models are typically associated with a directed acyclic graph (DAG) which represents the functional relationships between the variables of interest. In situations where the joint distribution is faithful, meaning that no additional conditional independences are generated by the joint distribution (Spirtes et al., 2000), the conditional independence properties of the variables can be read from the graph itself through a concept known as d-separation (Geiger et al., 1990). We will use d-separation as our primary tool for operating on the probabilistic expressions. The reader is assumed to be familiar with a number of graph theoretic concepts that are explained for example in (Koller and Friedman, 2009) and used throughout the paper.

Our simplification procedure is built on the definition of simplification sets, which is presented in Section 3. We continue by introducing a sound and complete simplification algorithm for probabilistic expressions defined in Section 2 for which these simplification sets exist. The algorithm takes as an input the expression to be simplified and the graph induced by the underlying causal model, and proceeds to construct a joint distribution of the variables contained in the expression by using the d-separation criteria. Higher level algorithms that use this simplification procedure are presented in Section 4. These include an algorithm for the simplification of a nested expression and an algorithm for the simplification of a quotient of two expressions. Section 5 contains examples on the application of these algorithms. We have also updated the causaleffect R-package to automatically apply these simplification procedures to causal effect expressions.

As a motivating example we present an expression of a causal effect given by the ID algorithm of Shpitser and Pearl (2006a) that can be simplified. The complete derivation of this effect can be found in Appendix C. The causal effect of X on Z_1, Z_2, Z_3 and Y is identifiable in the graph of Figure 1 and application of the ID algorithm gives

$$P(Z_1|Z_2,X)P(Z_3|Z_2)\frac{\sum_X P(Y|Z_2,X,Z_3,Z_1)P(Z_3|Z_2,X)P(X|Z_2)P(Z_2)}{\sum_{X,Y} P(Y|Z_2,X,Z_3,Z_1)P(Z_3|Z_2,X)P(X|Z_2)P(Z_2)}\times \sum_{X,Z_3,Y} P(Y|Z_2,X,Z_3,Z_1)P(Z_3|Z_2,X)P(X|Z_2)P(Z_2).$$

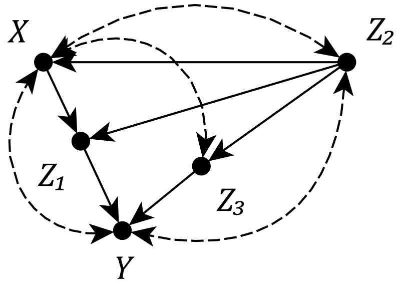

Figure 1: A graph for the introductory example on simplification.

It turns out that there exists a significantly simpler expression,

$$P(Z_1|Z_2, X)P(Z_2)\sum_{X} P(Y|Z_2, X, Z_3, Z_1)P(Z_3|Z_2, X)P(X|Z_2), \tag{1}$$

for the same causal effect. This expression can be obtained without any knowledge of the underlying model by using standard probability manipulations. However, this requires that a favorable choice is made for the ordering of the nodes of the graph in the ID algorithm. In the case that we had chosen an ordering where Z_1 precedes Z_3, the term for Z_3 would instead be $P(Z_3|Z_2, Z_1, X)$ and simplification would require knowledge about the underlying graph. We will take another look at this example later in Section 5 where we describe in detail how our procedure can be used to find expression (1).

Our simplification procedure is different from the well-known exact inference method of minimizing the amount of numerical computations when evaluating expressions for conditional and marginal distributions by changing the order of summations and multiplications in the expression. Variants of this method are known by different names depending on the context, such as Bayesian variable elimination (Koller and Friedman, 2009) and the sum-product algorithm (Bishop, 2006) which is a generalization of belief propagation (Pearl, 1988; Lauritzen and Spiegelhalter, 1988). Efficient computational methods exist for causal effects as well, such as (Shpitser et al., 2011). The general principle is the same in all of the variants, and no symbolic simplification is performed.

In our setting simplification can be defined explicitly but in general it is difficult to say what makes one expression simpler than another. Carette (2004) provides a formal definition for simplification in the context of Computer Algebra Systems (CAS) that operate on algebraic expressions. Modern CAS systems such as Mathematica (Wolfram Research Inc., 2015) and Maxima (Maxima, 2014) implement techniques for symbolic simplification. Bailey et al. (2014) and references therein discuss simplification techniques in CAS systems further. However to the best of our knowledge, the symbolic simplification procedures for probabilistic expressions described in this paper have neither been given previous attention nor implemented in any existing system.

2. Probabilistic Expressions

Every expression that we consider is defined in terms of a set of variables $\mathbf{W}$. As we are interested in probabilistic expressions, we also assume a joint probability distribution P for the variables of $\mathbf{W}$. The most basic of expressions are called atomic expressions which will be the main focus of this paper.

Definition 1 (Atomic expression) *Let* $\mathbf{W}$ *be a set of* p *discrete random variables and let* P *be any joint distribution of* $\mathbf{W}$*. An atomic expression is a pair*

$$A = A[\mathbf{W}] = \langle \mathbf{T}, \mathbf{S} \rangle,$$

where

1. $\mathbf{T}$ *is a set of pairs* $\{\langle V_1, \mathbf{C}_1 \rangle, \ldots, \langle V_n, \mathbf{C}_n \rangle\}$ *such that for each* V_i *and* $\mathbf{C}_i$ *it holds that* $V_i \in \mathbf{W}$, $\mathbf{C}_i \subseteq \mathbf{W}$, $V_i \notin \mathbf{C}_i$ *and* $V_i \neq V_j$ *for* $i \neq j$.
2. $\mathbf{S}$ *is a set* $\{S_1, \ldots, S_m\} \subseteq \mathbf{W}$ *such that for each* $i = 1, \ldots, m$ *it holds that* $S_i = V_j$ *for some* $j \in \{1, \ldots, n\}$.

The value of an atomic expression A *is*

$$P_A = \sum_{\mathbf{S}} \prod_{i=1}^{n} P(V_i | \mathbf{C}_i).$$

The probabilities $P(V_i|\mathbf{C}_i)$ are referred to as the terms of the atomic expression. A term $P(V_i|\mathbf{C}_i)$ is said to contain a variable V if $V_i = V$ or $V \in \mathbf{C}_i$. A term for a variable V refers to a term $P(V|\cdot)$. We also use the shorthand notation $V[A] := \{V_1, \ldots V_n\}$. As $\mathbf{S}$ is a set, we will only sum over a certain variable once. All variables are assumed to be univariate and discrete for clarity, but we may also consider multivariates and situations where some of the variables are continuous and the respective sums are interpreted as integrals instead.

As an example we will construct an atomic expression describing the following formula

$$\sum_{X} P(Y|Z_2, X, Z_3, Z_1) P(Z_3|Z_2, X) P(X|Z_2) P(Z_2),$$

which is a part of the motivating example in the introduction. We let $\mathbf{W} = \{X, Y, Z_1, Z_2, Z_3\}$, which is the set of nodes of the graph of Figure 1. The sets $\mathbf{T}$ and $\mathbf{S}$ can now be defined as

$$\{\langle Y, \{Z_2, X, Z_3, Z_1\}\rangle, \langle Z_3, \{Z_2, X\}\rangle, \langle X, \{Z_2\}\rangle, \langle Z_2, \emptyset \rangle\} \quad \text{and} \quad \{X\},$$

respectively. Next we define a more general probabilistic expression.

Definition 2 (Expression) *Let* $\mathbf{W}$ *be a set of* p *variables and let* P *be the joint distribution of* $\mathbf{W}$*. An expression is a triple*

$$B = B[\mathbf{W}, n, m] = \langle \mathbf{B}, \mathbf{A}, \mathbf{S} \rangle,$$

where

1. *$\mathbf{S}$ is a subset of $\mathbf{W}$.*
2. *For $m > 0$, $\mathbf{A}$ is a set of atomic expressions*
$$\{\langle \mathbf{T}_1, \mathbf{S}_1 \rangle, \ldots, \langle \mathbf{T}_m, \mathbf{S}_m \rangle\}.$$
If $m = 0$ then $\mathbf{A} = \emptyset$.
3. *For $n > 0$, $\mathbf{B}$ is a set of expressions*
$$\{B_1[\mathbf{W}_1, n_1, m_1], \ldots, B_n[\mathbf{W}_n, n_n, m_n]\}$$
such that $\mathbf{W}_i \subseteq \mathbf{W}$, $n_i < n, m_i < m$ for all $i = 1, \ldots, n$. If $n = 0$ then $\mathbf{B} = \emptyset$.

The value of an expression B is

$$P_B = \sum_{\mathbf{S}} \prod_{i=1}^{n} P_{B_i} \prod_{j=1}^{m} P_{A_j},$$

where an empty product should be understood as being equal to 1.

The recursive definition ensures the finiteness of the resulting expression by requiring that each sub-expression has fewer sub-expressions of their own than the expression above it. A single value might be shared by multiple expressions, as the terms of the product in the value of the expression are exchangeable. Expressions $B_1[\mathbf{W}, n_1, m_1]$ and $B_2[\mathbf{W}, n_2, m_2]$ are equivalent if their values P_{B_1} and P_{B_2} are equal for all P. Equivalence is defined similarly for atomic expressions. Every expression is formed by nested atomic expressions by definition. Because of this, we focus on the simplification of atomic expressions.

As an example we construct an expression for the causal effect formula (1). We define $\mathbf{W} := \{X, Y, Z_1, Z_2, Z_3\}$ and let the sets $\mathbf{B}$ and $\mathbf{S}$ be empty. We define the set $\mathbf{A}$ to consist of three atomic expressions A_1, A_2 and A_3 defined as follows

$$\begin{aligned}
A_1 &= \langle \{\langle Z_1, \{Z_2, X\}\rangle\}, \emptyset \rangle, \\
A_2 &= \langle \{\langle Z_2, \emptyset\}\rangle\}, \emptyset \rangle, \\
A_2 &= \langle \{\langle Y, \{Z_2, X, Z_3, Z_1\}\rangle, \langle Z_3, \{Z_2, X\}\rangle, \langle X, \{Z_2\}\rangle, \langle Z_2, \emptyset\rangle\}, \{X\}\rangle.
\end{aligned}$$

In the context of probabilistic graphical models, we are provided additional information about the joint distribution of the variables of interest in the form of a DAG. As we are concerned on the simplification of the results of causal effect derivations in such models, the general form of the atomic expressions can be further narrowed down by using the structure of the graph and the ordering of vertices called a topological ordering.

Definition 3 (Topological ordering) *Topological ordering π of a DAG $G = \langle \mathbf{W}, \mathbf{E} \rangle$ is an ordering of its vertices, such that if X is an ancestor of Y in G then $X < Y$ in π.*

The symbol V_j^π is used to denote the subset of vertices of G that are less than V_j in π. For sets we may define $\mathbf{V}^\pi$ to contain those vertices of G that are less than every vertex of $\mathbf{V}$ in π. Consider a DAG $G = \langle \mathbf{W}, \mathbf{E} \rangle$ and a topological ordering π of its vertices. We use the notation $\pi(\cdot)$ to denote indexing over the vertex set $\mathbf{W}$ of G in the ordering given by π, that is $V_{\pi(1)} > V_{\pi(2)} > \cdots > V_{\pi(m)}$ where $m = |\mathbf{W}|$. For any atomic expression $A[\mathbf{V}] = \langle \mathbf{T}, \mathbf{S} \rangle$ such that $\mathbf{V} \subseteq \mathbf{W}$ we also define the induced ordering ω. This ordering is an ordering of the variables in $\mathbf{V}$ such that if $X > Y$ in ω then $X > Y$ also in π. From now on in this paper, any indexing over the variables of an atomic expression will refer to the induced ordering of the set $\mathbf{V}$ when π is given, i.e $V_1 > V_2 > \cdots > V_n$ in ω. In other words, ω is obtained from π by leaving out variables that are not contained in A.

The ID algorithm performs the so-called C-component factorization. These components are subgraphs of the original graph where every node is connected by a path consisting entirely of bidirected edges. The resulting expressions of these factors serve as the basis for our simplification procedure.

Definition 4 (Topological consistency) *Let G' be a DAG with a subgraph $G = \langle \mathbf{W}, \mathbf{E} \rangle$ and let π be a topological ordering of the vertices of G. An atomic expression $A[\mathbf{W}] = \langle \mathbf{T}, \mathbf{S} \rangle$ is topologically consistent (or π-consistent for short) if*

$$An(V_i)_G \subseteq \mathbf{C}_i \subseteq V_i^{\pi} \text{ for all } i = 1, \ldots, n.$$

Here $An(V_i)_G$ denotes the ancestors of V_i in G. To motivate this definition we note that the outputs of the algorithms of Shpitser and Pearl (2006a,b) can always be represented by using products and quotients of topologically consistent atomic expressions. An expression is topologically consistent when every atomic expression contained by it is topologically consistent with respect to a topological ordering of a subgraph. We provide a proof for this statement in Appendix A. This also shows that any manual derivation of a causal effect can always be represented by a topologically consistent expression. The assumption that $An(V_i)_G \subseteq \mathbf{C}_i$ is not necessary for the simplification to be successful. This assumption is used to speed up the performance of our procedure in Section 3.

3. Simplification

Simplification in our context is the procedure of eliminating variables from the set of variables that are to be summed over in expressions. In atomic expressions, a successful simplification in terms of a single variable should result in another expression that holds the same value, but with the respective term eliminated and the variable removed from the summation. As we are interested in causal effects, we consider only simplification of topologically consistent atomic expressions.

Our approach to simplification is that the atomic expression has to represent a joint distribution of the variables present in the expression to make the procedure feasible. The question is whether the expression can be modified to represent a joint distribution. Before we can consider simplification, we have to define this property explicitly.

Definition 5 (Simplification sets) *Let G' be a DAG and let G be a subgraph of G' over a vertex set $\mathbf{W}$ with a topological ordering π. Let $A[\mathbf{W}] = \langle \mathbf{T}, \mathbf{S} \rangle$, where $\mathbf{T} = \{\langle V_1, \mathbf{C}_1, \rangle, \ldots, \langle V_n, \mathbf{C}_n \rangle\}$, be a π-consistent atomic expression and let $V_j \in \mathbf{S}$. Suppose that $V_{\pi(p)} = V_j$ and that $V_{\pi(q)} = V_1$ and let $\mathbf{M}$ be the set*

$$\{U \in \mathbf{W} \mid U \notin V[A], V_{\pi(q)} > U > V_{\pi(p)}\}.$$

If there exists a set $\mathbf{D} \subset V_j^\pi$ and the sets $\mathbf{E}_U \subseteq \mathbf{W}$ for all $U \in \mathbf{M}$ such that the conditional distribution of the variables $V_{\pi(p)}, \ldots, V_{\pi(q)}$ can be factorized as

$$P(V_{\pi(p)}, \ldots, V_{\pi(q)}|\mathbf{D}) = \prod_{U \in \mathbf{M}} P(U|\mathbf{E}_U) \prod_{V_i \geq V_j} P(V_i|\mathbf{C}_i), \tag{2}$$

and

$$(U \perp\!\!\!\perp V_j|\mathbf{E}_U \setminus \{V_j\})_{G'} \text{ for all } U \in \mathbf{M}. \tag{3}$$

then the sets $\mathbf{D}$ and $\mathbf{E}_U, U \in \mathbf{M}$ are the simplification sets of A with respect to V_j.

This definition is tailored for the next result that can be used to determine the existence of a simpler expression when simplification sets exist. Afterwards we will show how this result can be applied in practice via an example. The definition characterizes π-consistent atomic expressions that represent joint distributions. It is apparent that simplifications sets are not always unique, which can lead to different but still simpler expressions. Henceforth the next result considers simplification in terms of a single variable. The proof is available in Appendix B.

Theorem 6 (Simplification) *Let G' be a DAG and let G be a subgraph of G' over a vertex set $\mathbf{W}$ with a topological ordering π. Let $A[\mathbf{W}] = \langle \mathbf{T}, \mathbf{S} \rangle$ be a π-consistent atomic expression and let $\mathbf{D}$ and $\mathbf{E}_U, U \in \mathbf{M}$ be its simplification sets with respect to a variable $V_j \in \mathbf{S}$. Then there exist an expression $A'[\mathbf{W} \setminus \{V_j\}] = \langle \mathbf{T}', \mathbf{S}' \rangle$ such that $V_j \notin \mathbf{S}'$, $P_A = P_{A'}$ and no term in A' contains V_j.*

Note that even if $\mathbf{M} = \emptyset$ in Definition 5, the existence of simplification sets still requires that $\prod_{V_i \geq V_j} P(V_i|\mathbf{C}_i) = P(V_j, \ldots, V_1|\mathbf{D})$. In many cases there exists variables $U \in \mathbf{M}$ such that the expression does not contain a term for U. Condition (2) of Definition 5 guarantees that if these terms were contained in the expression it would represent a joint distribution. Our goal is thus to introduce these terms into the original expression temporarily, carry out the desired summation, and finally remove the added terms. This can only be achieved if the variables in the set $\mathbf{M}$ are conditionally independent of the variable currently being summed over, hence the assumption $(U \perp\!\!\!\perp V_j|\mathbf{E}_U \setminus \{V_j\})_{G'}$ of condition (3) of Definition 5.

We show how simplification sets can be used in practice to derive a simpler expression via an example. We consider the causal effect of $\{X, Z, W\}$ on Y in the graph G of Figure 2. The effect in question is identifiable and the ID algorithm readily gives atomic expression

$$\sum_{X,W} P(Y|X, W, Z)P(X|W)P(W).$$

We consider simplification sets with respect to $V_j = W$. The topological order is $W < X < Z < Y$. The atomic expression does not contain a term for Z so we have $\mathbf{M} = \{Z\}$. By noting that $(Z \perp\!\!\!\perp W|X)_G$ we are able to satisfy condition (3) of Definition 5. We can write

$$P(Y, Z, X, W) = P(Z|X, W)P(Y|X, W, Z)P(X|W)P(W),$$

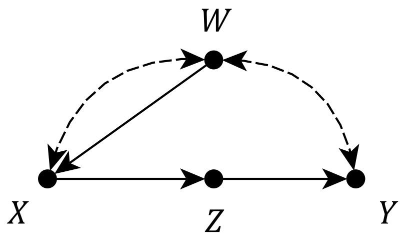

Figure 2: A graph G for the example on the use of simplification sets.

as required by condition (2) of Definition 5 by setting $\mathbf{E}_Z = \{X, W\}$. Thus, the simplification sets $\mathbf{D}$ and $\mathbf{E}_Z$ for the atomic expression with respect to W are $\emptyset$ and $\{X, W\}$, respectively. Finally, we obtain the simpler atomic expression by carrying out the summation over W:

$$\sum_X P(Y|X, Z)P(X).$$

Neither Definition 5 nor Theorem 6 provide a method to obtain simplification sets or to determine whether they exist in general. To solve this problem we present a simplification algorithm for π-consistent atomic expressions that operates by constructing simplification sets iteratively for each variable in the summation set.

Algorithm 1 always attempts to perform maximal simplification, meaning that as many variables of the set $\mathbf{S}$ are removed as possible. If the simplification in terms of the entire set $\mathbf{S}$ can not be completed, the intermediate result with as many variables simplified as possible is returned. If simplification in terms of specific variables or a subset is preferred, the set $\mathbf{S}$ should be defined accordingly.

The function SIMPLIFY takes three arguments: an atomic expression $A[\mathbf{W}]$ that is to be simplified, a graph G and a topological ordering π of its vertices. A is assumed to be π-consistent.

On line 10 the function INDEX.OF returns the corresponding index i of the term containing S_j. Since A is π-consistent, we only have to iterate through the variables $V_1, \ldots, V_j$ as the terms outside this range contain no relevant information about the simplification of V_j. The variables without a corresponding term in the atomic expression A are retrieved on line 11 by the function GET.MISSING. This function returns the set $\mathbf{M}$ of Definition 5 with respect to the current variable to be summed over.

In order to show that the term of A represent some joint distribution, we proceed in the order dictated by the topological ordering of the vertices. The sets $\mathbf{J}$ and $\mathbf{D}$ keep track of the variables that have been successfully processed and of the conditioning set of the joint term that was constructed on the previous iteration. Similarly, the sets $\mathbf{R}$ and $\mathbf{I}$ keep track of the variables and conditioning sets of the corresponding variables that the atomic expression does not originally contain a term for. Iteration through relevant terms begins on line 13. Next, we take a closer look at the function JOIN which is called next on line 14.

Here $\mathcal{P}(\cdot)$ denotes the power set, $\triangle$ denotes the symmetric difference and $An^*(\cdot)_G$ denotes the ancestors with the argument included. The function JOIN attempts to combine the joint

Algorithm 1 Simplification of an atomic expression $A = \langle \mathbf{T}, \mathbf{S} \rangle$ given graph G and topological ordering π.

```
function SIMPLIFY($A, G, \pi$)
    $j \leftarrow 0$
    while $j < |\mathbf{S}|$ do
        $B \leftarrow A$
        $\mathbf{J} \leftarrow \emptyset$
        $\mathbf{D} \leftarrow \emptyset$
        $\mathbf{R} \leftarrow \emptyset$
        $\mathbf{I} \leftarrow \emptyset$
        $j \leftarrow j + 1$
        $i \leftarrow$ INDEX.OF($A, j$)
        $\mathbf{M} \leftarrow$ GET.MISSING($A, G, j$)
        $k \leftarrow 1$
        while $k \leq i$ do
            $\langle \mathbf{J}_{\text{new}}, \mathbf{D}_{\text{new}}, \mathbf{R}_{\text{new}} \rangle \leftarrow$ JOIN($\mathbf{J}, \mathbf{D}, V_k, \mathbf{C}_k, S_j, \mathbf{M}, G, \pi$)
            if $\mathbf{J}_{\text{new}} \subseteq \mathbf{J}$ then
                break
            else
                $\mathbf{J} \leftarrow \mathbf{J}_{\text{new}}$
                $\mathbf{D} \leftarrow \mathbf{D}_{\text{new}}$
                if $\mathbf{R}_{\text{new}} \neq \emptyset$ then
                    $\mathbf{R} \leftarrow \mathbf{R} \cup \mathbf{R}_{\text{new}}$
                    $\mathbf{I} \leftarrow \mathbf{I} \cup \{\mathbf{D}\}$
                    $\mathbf{M} \leftarrow \mathbf{M} \setminus \mathbf{R}_{\text{new}}$
                else
                    $k \leftarrow k + 1$
        if $k = i + 1$ then
            $A_{\text{new}} \leftarrow$ FACTORIZE($\mathbf{J}, \mathbf{D}, \mathbf{R}, \mathbf{I}, A$)
            if $A_{\text{new}} = A$ then
                $A \leftarrow B$
            else
                $A \leftarrow A_{\text{new}}$
                $\mathbf{S} \leftarrow \mathbf{S} \setminus \{S_j\}$
                $j \leftarrow 0$
    return $A$
```

term $P(\mathbf{J}|\mathbf{D})$, obtained from the previous iteration steps, with the term $P(V|\mathbf{C}) := P(V_k|\mathbf{C}_k)$ of the current iteration step. d-separation statements of G are evaluated to determine whether this can be done. In practice this means finding a suitable subset $\mathbf{P}_i$ of $\mathbf{G}$, where $\mathbf{G} \cup An(V)_G$ is the largest possible conditioning set of the new combined term. The set $\mathbf{G}$ is computed on line on line 4 of Algorithm 2. A valid subset $\mathbf{P}_i$ satisfies $P(\mathbf{J}|\mathbf{D}) = P(\mathbf{J}|An^*(V)_G, \mathbf{P}_i)$ and $P(V|\mathbf{C}) = P(V|An(V)_G, \mathbf{P}_i)$ which allow us to write the product $P(\mathbf{J}|\mathbf{D})P(V|\mathbf{C})$ as $P(\mathbf{J}, V|An(V)_G, \mathbf{P}_i)$.

Algorithm 2 Construction of the joint distribution of the set $\mathbf{J}$ and a variable V given their conditional sets $\mathbf{D}$ and $\mathbf{C}$ using d-separation criteria in G. S is the current summation variable, $\mathbf{M}$ is the set of variables not contained in the expression and π is a topological ordering.

```
 1: function JOIN($\mathbf{J}, \mathbf{D}, V, \mathbf{C}, S, \mathbf{M}, G, \pi$)
 2:     if $\mathbf{J} = \emptyset$ then
 3:         return $\langle \{V\}, \mathbf{C}, \emptyset \rangle$
 4:     $\mathbf{G} \leftarrow \mathbf{J}^{\pi} \setminus An^{*}(V)_G$
 5:     $\mathbf{P} \leftarrow \mathcal{P}(\mathbf{G})$
 6:     $n \leftarrow |\mathbf{P}|$
 7:     for $i = 1 : n$ do
 8:         $\mathbf{A} \leftarrow (An^{*}(V)_G \cup \mathbf{P}_i) \triangle \mathbf{D}$
 9:         $\mathbf{B} \leftarrow (An(V)_G \cup \mathbf{P}_i) \triangle \mathbf{C}$
10:         if $(\mathbf{J} \perp\!\!\!\perp \mathbf{A} | \mathbf{D} \setminus \mathbf{A})_G$ and $(V \perp\!\!\!\perp \mathbf{B} | \mathbf{C} \setminus \mathbf{B})_G$ then
11:             return $\langle \mathbf{J} \cup \{V\}, (An(V)_G \cup \mathbf{P}_i), \emptyset \rangle$
12:     if $\mathbf{M} \neq \emptyset$ then
13:         for $M' \in \mathbf{M}$ do
14:             if $M' \in \mathbf{D}, M' \notin \mathbf{C}$ then
15:                 $\langle \mathbf{J}_{\text{new}}, \mathbf{D}_{\text{new}}, \mathbf{R} \rangle \leftarrow$ INSERT($\mathbf{J}, \mathbf{D}, M', S, G, \pi$)
16:                 if $\mathbf{J} \subset \mathbf{J}_{\text{new}}$ then
17:                     return $\langle \mathbf{J}_{\text{new}}, \mathbf{D}_{\text{new}}, \mathbf{R} \rangle$
18:     return $\langle \mathbf{J}, \mathbf{D}, \emptyset \rangle$
```

In order to find this valid subset, we compute the sets $\mathbf{A}$ and $\mathbf{B}$ for each candidate on lines 8 and 9. These sets characterize the necessary change in the conditioning sets of the terms $P(\mathbf{J}|\mathbf{D})$ and $P(V|\mathbf{C})$ that would enable a joint term to be formed by these two terms. The validity of the candidate set is finally checked on line 10 which determines if the necessary change is allowed by d-separation criteria in the graph G. If no valid subset $\mathbf{P}_i$ can be found, we can still attempt to insert a missing variable of $\mathbf{M}$ by calling INSERT. If this does not succeed either, the original sets $\mathbf{J}$ and $\mathbf{D}$ are returned, which instructs SIMPLIFY to terminate simplification in terms of V_j and attempt simplification in the next variable.

A special case where the first variable of the joint distribution forms $P(\mathbf{J}, \mathbf{D})$ alone is processed on line 2 of Algorithm 2. In this case, we have an immediate result without having to iterate through the subsets of $\mathbf{G}$. The formulation of the set $\mathbf{G}$ ensures that the resulting factorization is π-consistent if it exists. Knowing that the ancestral set $An(V)_G$ has to be a subset of the new conditioning set also greatly reduces the amount of subsets we have to iterate through. In a typical situation, the size of $\mathbf{P}$ is not very large. Let us now inspect the insertion procedure in greater detail.

In essence, the function INSERT is a simpler version of JOIN, because the only restriction on the conditioning set of M' is imposed by the conditioning set of $\mathbf{J}$ and the fact that M' has to be conditionally independent of the current variable S to be summed over. If JOIN or INSERT was unsuccessful in forming a new joint distribution, we have that $\mathbf{J}_{\text{new}} \subset \mathbf{J}$. In this

Algorithm 3 Insertion of variable M' into the joint term $P(\mathbf{J}|\mathbf{D})$ using d-separation criteria in G. S is the current summation variable and π is a topological ordering.

```
function INSERT($\mathbf{J}, \mathbf{D}, M', S, G, \pi$)
    $\mathbf{G} \leftarrow \mathbf{J}^{\pi} \setminus An^*(M')_G$
    $n \leftarrow |\mathbf{G}|$
    for $i = 1 : n$ do
        $\mathbf{A} \leftarrow (An^*(M')_G \cup \mathbf{P}_i) \triangle \mathbf{D}$
        $\mathbf{B} \leftarrow (An(M')_G \cup \mathbf{P}_i)$
        if $(\mathbf{J} \perp\!\!\!\perp \mathbf{A}|\mathbf{D} \setminus \mathbf{A})_G$ and $(M' \perp\!\!\!\perp \mathbf{S}|\mathbf{B} \setminus \mathbf{S})_G$ then
            return $\langle \mathbf{J} \cup \{M'\}, (An^*(M')_G \cup \mathbf{P}_i), \{M'\} \rangle$
    return $\langle \mathbf{J}, \mathbf{D}, \emptyset \rangle$
```

case simplification in terms of the current variable cannot be completed. If we have that $\mathbf{J}_{\text{new}} \not\subset \mathbf{J}$ the iteration continues.

Together the functions JOIN and INSERT capture the two conditions of Definition 5. They are essentially two variations of the underlying procedure of determining whether the terms of the atomic expression actually represent a joint distribution. The only difference is that JOIN is called when we are processing terms that already exist in the expression, and INSERT is called when there are variables without corresponding terms in the expression, that is the set $\mathbf{M}$ of Definition 5 is not empty.

If the innermost while-loop of Algorithm 1 succeeded in iterating through the relevant variables, we are ready to complete the simplification process in terms of S_j. We carry out the summation over S_j which results in $P(\mathbf{J} \setminus \{V_i\}|\mathbf{D})$. This is done on line 27 by calling FACTORIZE$(\mathbf{J}, \mathbf{D}, \mathbf{R}, \mathbf{I}, A)$ which checks whether the joint term $P(\mathbf{J} \setminus \{V_i\}|\mathbf{D})$ can be factorized back into a product of terms. In practice this means that if the function succeeds, it will return an atomic expression obtained by removing each inserted term $P(R|\mathbf{I}_R)$ such that $R \in \mathbf{R}$ and $\mathbf{I}_R \in \mathbf{I}$ from atomic expression A. The status of the atomic expression is updated on lines 31 and 32 to reflect this. If the function fails, it will return A unchanged.

If the innermost while-loop did not iterate completely through the relevant variables, the simplification was not successful in terms of S_j at this point. In this case we reset A to its original state on line 29 and attempt simplification in terms of the next variable. If there are no further variables to be eliminated, the outermost while-loop will also terminate. In the next theorem, we show that Algorithm 1 is both sound and complete in terms of simplification sets. The proof for the theorem can be found in Appendix D.

Theorem 7 *Let G' be a DAG and let G be a subgraph of G' over a vertex set $\mathbf{W}$ with a topological ordering π. Let $A[\mathbf{W}] = \langle \mathbf{T}, \{V_j\} \rangle$ be a π-consistent atomic expression. Then if* SIMPLIFY(A, G, π) *succeeds, it has constructed a collection of simplification sets of A with respect to V_j. Conversely, if there exists a collection of simplifications sets of A with respect to V_j, then* SIMPLIFY(A, G, π) *will succeed.*

4. High Level Algorithms

In this section, we present an algorithm to simplify all atomic expressions in the recursive stack of an expression. We will also provide a simple procedure to simplify quotients defined by two expressions: one representing the numerator and another representing the denominator. In some cases it is also possible to eliminate the denominator by subtracting common terms. First, we present a general algorithm to simplify topologically consistent expressions.

Algorithm 4 Recursive wrapper for the simplification of an expression $B = \langle \mathbf{B}, \mathbf{A}, \mathbf{S} \rangle$ given graph G and topological ordering π.

```
 1: function DECONSTRUCT($B, G, \pi$)
 2:     $\mathbf{R} \leftarrow \emptyset$
 3:     for $Y \in \mathbf{A}$ do
 4:         $\langle \{ \langle V_1, \mathbf{C}_1 \rangle, \ldots, \langle V_n, \mathbf{C}_n \rangle \}, \mathbf{S}_Y \rangle \leftarrow$ SIMPLIFY($Y, G, \pi$)
 5:         if $\mathbf{S}_Y = \emptyset$ then
 6:             $\mathbf{A} \leftarrow \mathbf{A} \cup (\bigcup_{i=1}^{n} \{ \langle \{ \langle V_i, \mathbf{C}_i \rangle \}, \emptyset \rangle \})$
 7:     for $\langle \mathbf{B}_X, \mathbf{A}_X, \mathbf{S}_X \rangle \in \mathbf{B}$ do
 8:         $\langle \mathbf{B}_X, \mathbf{A}_X, \mathbf{S}_X \rangle \leftarrow$ DECONSTRUCT($\langle \mathbf{B}_X, \mathbf{A}_X, \mathbf{S}_X \rangle, G$)
 9:         if $\mathbf{B}_X = \emptyset$ and $\mathbf{S}_X = \emptyset$ then
10:             $\mathbf{R} \leftarrow \mathbf{R} \cup \{ \langle \mathbf{B}_X, \mathbf{A}_X, \mathbf{S}_X \rangle \}$
11:             $\mathbf{A} \leftarrow \mathbf{A} \cup \mathbf{A}_X$
12:     $\mathbf{B} \leftarrow \mathbf{B} \setminus \mathbf{R}$
13:     return $\langle \mathbf{B}, \mathbf{A}, \mathbf{S} \rangle$
```

Algorithm 4 begins by simplifying all atomic expressions contained in the expressions. If an atomic expression contains no summations after the simplification but does contain multiple terms, each individual term is converted into an atomic expression of their own. After this, we iterate through all sub-expressions contained in the expression. The purpose of this is to carry out the simplification of every atomic expression in the stack and collect the results into as few atomic expressions as possible. First, we traverse to the bottom of the stack on line 8 by deconstructing sub-expressions until they have no sub-expressions of their own. Afterwards, it must be the case that $\langle \mathbf{B}_X, \mathbf{A}_X, \mathbf{S}_X \rangle$ consists of atomic sub-expressions only.

If $\langle \mathbf{B}_X, \mathbf{A}_X, \mathbf{S}_X \rangle$ contains no summations on line 9 then the atomic expressions contained in this expression do not require an additional expression to contain them, but can instead be transferred to be a part of the expression above the current one in the recursive stack. On line 6 we lift the atomic expressions contained in the atomic sub-expressions up to the current recursion stage.

There is no guarantee, that the resulting atomic expression is still π-consistent after this procedure. The function DECONSTRUCT operates on the principle of simplifying as many atomic expressions as possible, combining the results into new atomic expressions and simplifying them once more. We do not claim that this procedure is complete in a sense that Algorithm 4 would always find the simplest representation for a given expression.

This method in nonetheless sound and finds drastically simpler expressions in almost every situation where such an expression exists.

We may also consider quotients often formed by deriving conditional distributions. For this purpose we need a subroutine to extract terms from atomic sub-expression that are independent of the summation index, that is $V_i \not\in \mathbf{S}$ and $\mathbf{C}_i \cap \mathbf{S} = \emptyset$.

Algorithm 5 Extraction of terms independent of the summation indices from a expression $B = \langle \mathbf{B}, \mathbf{A}, \mathbf{S} \rangle$ given graph G and topological ordering π.

```
 1: function EXTRACT($B, G, \pi$)
 2:     $B \leftarrow$ DECONSTRUCT($B, G, \pi$)
 3:     if $\mathbf{S} = \emptyset$ then
 4:         for $X \in \mathbf{B}$ do
 5:             $X \leftarrow$ EXTRACT($X, G, \pi$)
 6:         for $\langle \mathbf{T}_A, \mathbf{S}_A \rangle \in \mathbf{A}$ do
 7:             if $\mathbf{S}_A \neq \emptyset$ then
 8:                 $\mathbf{A}_E \leftarrow \emptyset$
 9:                 $\mathbf{R} \leftarrow \emptyset$
10:                 for $\langle V, \mathbf{C} \rangle \in \mathbf{T}_A$ do
11:                     if $V \not\in \mathbf{S}_A$ and $\mathbf{C} \cap \mathbf{S}_A = \emptyset$ then
12:                         $\mathbf{A}_E \leftarrow \mathbf{A}_E \cup \{\langle \{\langle V, \mathbf{C} \rangle\}, \emptyset \rangle\}$
13:                         $\mathbf{R} \leftarrow \mathbf{R} \cup \{\langle V, \mathbf{C} \rangle\}$
14:                 $\mathbf{A} \leftarrow \mathbf{A} \cup \mathbf{A}_E$
15:                 $\mathbf{T}_A \leftarrow \mathbf{T}_A \setminus \mathbf{R}$
16:     else
17:         $\mathbf{A}_E \leftarrow \emptyset$
18:         $\mathbf{R} \leftarrow \emptyset$
19:         for $\langle \mathbf{T}_A, \mathbf{S}_A \rangle \in \mathbf{A}$ do
20:             if $\mathbf{S}_A = \emptyset$ then
21:                 $\mathbf{T}_A^{(1)} \leftarrow \emptyset$
22:                 $\mathbf{T}_A^{(2)} \leftarrow \emptyset$
23:                 for $\langle V, \mathbf{C} \rangle \in \mathbf{T}_A$ do
24:                     $\mathbf{T}_A^{(1)} \leftarrow \mathbf{T}_A^{(1)} \cup \{V\}$
25:                     $\mathbf{T}_A^{(2)} \leftarrow \mathbf{T}_A^{(2)} \cup \mathbf{C}$
26:                 if $\mathbf{T}_A^{(1)} \cap \mathbf{S} = \emptyset$ and $\mathbf{T}_A^{(2)} \cap \mathbf{S} = \emptyset$ then
27:                     $\mathbf{A}_E \leftarrow \mathbf{A}_E \cup \{\langle \mathbf{T}_A, \mathbf{S}_A \rangle\}$
28:                     $\mathbf{R} \leftarrow \mathbf{R} \cup \{\langle \mathbf{T}_A, \mathbf{S}_A \rangle\}$
29:         $\mathbf{A} \leftarrow \mathbf{A} \setminus \mathbf{R}$
30:         $\mathbf{B}_E \leftarrow \{B\}$
31:         return $\langle \mathbf{B}_E, \mathbf{A}_E, \emptyset \rangle$
```

The procedure of Algorithm 5 is rather straightforward. First, we attempt to simplify B by using DECONSTRUCT on line 2. Next, we simply recurse as deep as possible without encountering a sum in an expression. If a sum is encountered, extraction is attempted.

On any stage where a sum was not encountered, we may still have atomic sub-expression that contain sums. Because the recursion had reached this far, we know that there are no summations above them in the stack, so we can attempt extraction on them as well.

Algorithm 6 Simplification of a quotient P_{B_1}/P_{B_2} given by the values of two expressions $B_1 = \langle \mathbf{B}_1, \mathbf{A}_1, \mathbf{S}_1 \rangle$ and $B_2 = \langle \mathbf{B}_2, \mathbf{A}_2, \mathbf{S}_2 \rangle$ given graph G and topological ordering π.

```
 1: function q-SIMPLIFY($B_1, B_2, G, \pi$)
 2:     $B_1 \leftarrow$ EXTRACT($B_1, G, \pi$)
 3:     $B_2 \leftarrow$ EXTRACT($B_2, G, \pi$)
 4:     if $\mathbf{S}_1 \neq \emptyset$ or $\mathbf{S}_2 \neq \emptyset$ then
 5:         return $\langle B_1, B_2 \rangle$
 6:     $i \leftarrow 1$
 7:     while $i \leq |\mathbf{B}_1|$ and $|\mathbf{B}_1| > 0$ and $|\mathbf{B}_2| > 0$ do
 8:         for $j = 1 : |\mathbf{B}_2|$ do
 9:             if $B_{1i} = B_{2j}$ then
10:                 $\mathbf{B}_1 \leftarrow \mathbf{B}_1 \setminus \{B_{1i}\}$
11:                 $\mathbf{B}_2 \leftarrow \mathbf{B}_2 \setminus \{B_{2j}\}$
12:                 $i \leftarrow 0$
13:                 break
14:         $i \leftarrow i + 1$
15:     $i \leftarrow 1$
16:     while $i \leq |\mathbf{A}_1|$ and $|\mathbf{A}_1| > 0$ and $|\mathbf{A}_2| > 0$ do
17:         for $j = 1 : |\mathbf{A}_2|$ do
18:             if $A_{1i} = A_{2j}$ then
19:                 $\mathbf{A}_1 \leftarrow \mathbf{A}_1 \setminus \{A_{1i}\}$
20:                 $\mathbf{A}_2 \leftarrow \mathbf{A}_2 \setminus \{A_{2j}\}$
21:                 $i \leftarrow 0$
22:                 break
23:         $i \leftarrow i + 1$
24:     return $\langle B_1, B_2 \rangle$
```

Algorithm 6 takes two expressions, B_1 and B_2, and removes any sub-expressions and atomic sub-expressions that are shared by B_1 and B_2. This is of course only feasible when the summation sets are empty for both B_1 and B_2. This condition is checked on line 4.

5. Examples

In this section we present examples of applying the algorithms of the previous sections. We denote line number y of algorithm x with Ax:y. We begin with a simple example on the necessity of the INSERT procedure in graph G of Figure 3.

The causal effect of W on X is identifiable in this graph, and expression

$$\sum_{Z,Y} P(Y)P(Z|Y)P(X|W,Z,Y)$$

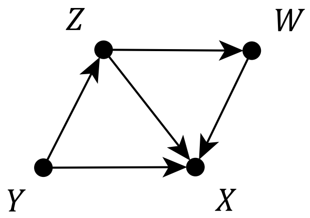

Figure 3: A graph G for the example on the necessity of the insertion procedure.

is obtained by direct application of the ID algorithm or by the truncated factorization formula for causal effects in Markovian models (Pearl, 2009). We let A be this atomic expression. The topological ordering π is $X > W > Z > Y$ and $\mathbf{M} = \{W\}$. The call to $\textsc{simplify}(A, G, \pi)$ will first attempt simplification in terms of Z, by calling

$$\textsc{join}(\emptyset, \emptyset, X, \{W, Z, Y\}, Z, \{W\}, G, \pi),$$

which results in $\langle X, \{W, Z, Y\}, \emptyset \rangle$. At the second call

$$\textsc{join}(\{X\}, \{W, Z, Y\}, Z, Y, Z, \{W\}, G, \pi)$$

we already run into trouble since we cannot find a conditioning set that would allow Z to be joined with $\{X\}$. However, since $\mathbf{M}$ is non-empty and $W \in \{W, Z, Y\}$ and $W \notin \{Z\}$ this means that the next call is

$$\textsc{insert}(\{X\}, \{W, Z, Y\}, W, Z, G, \pi).$$

Insertion fails in this case, as one can see from the fact that no conditioning set exists that would make W conditionally independent of Z. Thus we recurse back to join and back to simplify and end up on line A1:15 which breaks out of the while-loop. Thus A cannot be simplified in terms of Z. Simplification is attempted next in terms of Y. The first two calls are in this case

$$\textsc{join}(\emptyset, \emptyset, X, \{W, Z, Y\}, Y, \{W\}, G, \pi),$$

$$\textsc{join}(\{X\}, \{W, Z, Y\}, Z, \{Y\}, Y, \{W\}, G, \pi),$$

and in the second call we run into trouble again and have to attempt insertion

$$\textsc{insert}(\{X\}, \{W, Z, Y\}, W, Y, G, \pi).$$

This time we find that we can add a term for W which is $P(W|Z, Y)$ because $(W \perp\!\!\!\perp Y|Z)_G$. The other calls to join also succeed and we can write the value of A as

$$\frac{\sum_{Z,Y} P(Y)P(Z|Y)P(W|Y, Z)P(X|W, Z, Y)}{P(W|Z)}.$$

and complete the summation in terms of Y. After the call to factorize we are left with the final expression

$$\sum_{Z} P(X|W, Z)P(Z).$$

We continue by considering again graph G depicted in Figure 1. The topological ordering π is $Y > Z_1 > Z_3 > X > Z_2$. Atomic expression A_1 given by

$$\sum_{X,Y} P(Y|Z_2, X, Z_3, Z_1)P(Z_3|Z_2, X)P(X|Z_2)P(Z_2),$$

is a part of the expression to be simplified.

We will first simplify A_1 and take a closer look at how the function join operates. The call to simplify(A_1, G, π) will attempt simplification in terms of the set $\{X, Y\}$ in the ordering that agrees with the topological ordering π, which is (Y, X). After initializing the required sets, we find the index of the term with Y as a variable on line 10. There is one missing variable, Z_1, so $\mathbf{M} = \{Z_1\}$ as returned by get.missing on line A1:11. The first call to join results in $\langle Y, \{Z_2, X, Z_3, Z_1\}, \emptyset\rangle$, because line A2:3 is triggered. Condition on line A1:15 is not satisfied since $\mathbf{J}_{\text{new}} = \{Y\} \not\subseteq \emptyset = \mathbf{J}$. Thus we update the status of $\mathbf{J}$ and $\mathbf{D}$ on lines A1:18 and A1:19. Since $\mathbf{R}_{\text{new}} = \emptyset$ on line A1:20 we do not have to update the status of $\mathbf{R}, \mathbf{I}$ and $\mathbf{M}$ on lines A1:21, A1:22 and A1:23. The innermost while-loop is now complete and we call factorize on line A1:27 which succeeds in removing the term $P(Y|Z_2, X, Z_3, Z_1)$ by completing the sum. Now we update the status of the atomic expression on line A1:31 and remove Y from the set of variables to be summed over on line A1:32. The resulting value of the expression at this point is

$$\sum_{X} P(Z_3|Z_2, X)P(X|Z_2)P(Z_2).$$

Next, the summation in terms of X is attempted. join is once again successful, because Z_3 is the first variable to be joined and line A2:3 is triggered. Next we attempt to join the terms $P(Z_3|Z_2, X)$ and $P(X|Z_2)$. Computation of the set $\mathbf{G}$ on line A2:4 results in

$$\{Z_3\}^\pi \setminus An^*(X)_G = \{X, Z_2\} \setminus \{X, Z_2\} = \emptyset.$$

The power set computed on line A2:5 contains only the empty set. For $\mathbf{P}_1 = \emptyset$ we have

$$A = (An(X)^*_G \cup \mathbf{P}_1)\triangle\mathbf{D} = (\{X, Z_2\} \cup \emptyset)\triangle\{X, Z_2\} = \emptyset$$

on line 8, and

$$B = (An(X)_G \cup \mathbf{P}_1)\triangle\mathbf{C} = (\{Z_2\} \cup \emptyset)\triangle\{Z_2\} = \emptyset$$

on line 9. The condition on line A2:10 evaluates to true and we return with $\langle\{Z_3, X\}, \{Z_2\}, \emptyset\rangle$. The innermost while-loop terminates allowing the summation over X to be performed. The function factorize provides us with the final expression

$$P(Z_3|Z_2)P(Z_2). \tag{4}$$

Next, we will consider the full example and see how q-simplify is applied. Using the ID algorithm we obtain the causal effect of X on Z_1, Z_2, Z_3 and Y in graph G of Figure 1 and it is

$$P(Z_1|Z_2, X)P(Z_3|Z_2)\frac{\sum_X P(Y|Z_2, X, Z_3, Z_1)P(Z_3|Z_2, X)P(X|Z_2)P(Z_2)}{\sum_{X,Y} P(Y|Z_2, X, Z_3, Z_1)P(Z_3|Z_2, X)P(X|Z_2)P(Z_2)}\times$$
$$\sum_{X,Z_3,Y} P(Y|Z_2, X, Z_3, Z_1)P(Z_3|Z_2, X)P(X|Z_2)P(Z_2).$$

We will represent this as a quotient of expression using Definition 2. Let A_1 be the atomic expression of the previous example and let A_2 also be an atomic expression given by

$$\sum_{X} P(Y|Z_2, X, Z_3, Z_1)P(Z_3|Z_2, X)P(X|Z_2)P(Z_2),$$

which is essentially the same as A_1, but with the variable Y removed from the summation set $\mathbf{S}$. Similarly, we let A_3 be an atomic expression given by

$$\sum_{X,Z_3,Y} P(Y|Z_2, X, Z_3, Z_1)P(Z_3|Z_2, X)P(X|Z_2)P(Z_2).$$

We also define the atomic expressions A_4 with the value $P(Z_3|Z_2)$ and A_5 with the value $P(Z_1|Z_2, X)$. Now, we define two expressions B_1 and B_2 for the quotient P_{B_1}/P_{B_2} as follows:

$$B_1 = \langle \emptyset, \{A_2, A_3, A_4, A_5\}, \emptyset \rangle, \quad B_2 = \langle \emptyset, \{A_1\}, \emptyset \rangle.$$

We now call q-simplify(B_1, B_2, G, π). First, we must trace the calls to extract for both expressions on lines A6:2 and A6:3. For B_1 and B_2 this immediately results in a call to deconstruct on line A5:2. First, the function applies simplify to each atomic expression contained in the expressions on line A4:4.

Let us first consider the simplification of A_2. As before with A_1, we have that join first succeeds in forming $\langle Y, \{Z_2, X, Z_3, Z_1\}, \emptyset \rangle$, but this time Y is not in the summation set, so we continue. Next, the algorithm attempts to join $P(Y|Z_2, X, Z_3, Z_1)$ with $P(Z_3|Z_2, X)$. The set $\mathbf{G}$ is defined as

$$\{Y\}^{\pi} \setminus An^*(Z_3)_G = \{Z_3, Z_1, X, Z_2\} \setminus \{Z_3, Z_2\} = \{Z_1, X\}$$

and its subsets are $\{Z_1, X\}$, $\{Z_1\}$, $\{X\}$ and $\emptyset$. For the first subset $\mathbf{P}_1 = \emptyset$ we have that

$$A = (An^*(Z_3) \cup \mathbf{P}_1) \triangle \mathbf{D} = \{Z_2, Z_3\} \triangle \{Z_2, X, Z_3, Z_1\} = \{X, Z_1\}$$

and since $(Y \not\perp\!\!\!\perp X, Z_1|Z_3, Z_2)_G$ the condition on line A2:10 is not satisfied. We continue with $\mathbf{P}_2 = \{X\}$ and obtain

$$A = (An^*(Z_3) \cup \mathbf{P}_2) \triangle \mathbf{D} = \{X, Z_2, Z_3\} \triangle \{Z_2, X, Z_3, Z_1\} = \{Z_1\}$$

and since $(Y \not\perp\!\!\!\perp Z_1|X, Z_3, Z_2)_G$ the condition on line A2:10 is still not satisfied. Next, for $\mathbf{P}_3 = \{Z_1\}$ we have

$$A = (An^*(Z_3) \cup \mathbf{P}_3) \triangle \mathbf{D} = \{Z_2, Z_3, Z_1\} \triangle \{Z_2, X, Z_3, Z_1\} = \{X\}$$

and since $(Y \not\perp\!\!\!\perp X|Z_1, Z_3, Z_2)_G$ the condition on line A2:10 is again, not satisfied. Finally, for $\mathbf{P}_4 = \{Z_1, X\}$ we have

$$A = (An^*(Z_3) \cup \mathbf{P}_4) \triangle \mathbf{D} = \{Z_2, X, Z_3, Z_1\} \triangle \{Z_2, X, Z_3, Z_1\} = \{X\}$$

and

$$B = (An(Z_3) \cup \mathbf{P}_4) \triangle \mathbf{C} = \{Z_2, X, Z_1\} \triangle \{Z_2, X\} = \{Z_1\}.$$

Both conditions on line A2:10 are now satisfied ny noting that $(Z_3 \perp\!\!\!\perp Z_1 | X, Z_2)_G$. Afterwards we obtain

$$P(Y|Z_2, X, Z_3, Z_1)P(Z_3|Z_2, X) = P(Y, Z_3|Z_1, Z_2, X)$$

and continue in an attempt to join the term $P(X|Z_2)$ with this result. The set $\mathbf{G}$ is now defined as

$$\{Y, Z_3\}^{\pi} \setminus An^*(X)_G = \{Z_1, X, Z_2\} \setminus \{X, Z_2\} = \{Z_1\}$$

and its subsets are $\{Z_1\}$ and $\emptyset$. Starting with $\mathbf{P}_1 = \emptyset$ we have that

$$A = (An^*(X) \cup \mathbf{P}_1) \triangle \mathbf{D} = \{X, Z_2\} \triangle \{Z_1, Z_2, X\} = \{Z_1\}$$

and since $(Y, Z_3 \not\!\perp\!\!\!\perp Z_1 | X, Z_2)_G$ the condition on line A2:10 is not satisfied. Continuing with $\mathbf{P}_2 = \{Z_1\}$ we have

$$B = (An(X) \cup \mathbf{P}_2) \triangle \mathbf{C} = \{Z_2, Z_1\} \triangle \{Z_2\} = \{Z_1\}.$$

Again, the condition on line A2:10 is not satisfied by noting that $(X \not\!\perp\!\!\!\perp Z_1 | Z_2)_G$. We have exhausted the possible subsets, which means that we enter the loop on line A2:13 since the set $\mathbf{M} = \{Z_1\}$ is not empty of line A2:12.

In this case INSERT is called to bring Z_1 into the expression because $Z_1 \in \mathbf{D} = \{Z_1, Z_2, X\}$ and $Z_1 \notin \mathbf{C} = \{Z_2\}$. The set $\mathbf{G}$ is constructed on line A3:2 and it is

$$\mathbf{J}^{\pi} \setminus An^*(Z_1)_G = \{Y, Z_3\}^{\pi} \setminus \{X, Z_1, Z_2\} = \emptyset.$$

For the only subset $\mathbf{P}_1 = \emptyset$ we have

$$B = (An(Z_1)_G \cup \mathbf{P}_1) = \{X, Z_2\}$$

on line A3:6, and since $(Z_1 \not\!\perp\!\!\!\perp X | Z_2)_G$ the condition on line A3:7 is not satisfied and we return with $\langle \mathbf{J}, \mathbf{D}, \emptyset \rangle$ unchanged on line 9 of Algorithm 3, which causes JOIN to also return with the same output on line A3:18. The condition on line A1:15 is now satisfied and we cannot simplify A_2.

The atomic expression A_3 can be simplified. First, Y is eliminated exactly as it was removed from A_1. Following the same principle we can see that whenever a variable in the summation set is the largest one in the topological order of the variables contained in the atomic expression, it will be removed successfully. From this we obtain that the value of A_3 is in fact simply $P(Z_2)$. Let us call the atomic expression with this value E, that is $P_E = P(Z_2)$. The atomic expression A_1 can also be simplified, and its value is given by (4). Furthermore, since this value is made of two product terms, it is split into two atomic expressions respectively. Let these be called D_1 and D_2 such that $P_{D_1} = P(Z_3|Z_2)$ and $P_{D_2} = P(Z_2)$.

Applying SIMPLIFY to A_4 and A_5 simply returns the original expressions, since they do not contain any summations and the loop on line A1:3 is never entered. The set of atomic expressions is afterwards updated on line A4:6. Neither B_1 nor B_2 contain any sub-expressions or summations on line A4:9, so DECONSTRUCT(B_1, G, π) returns $\langle \emptyset, \{A_2, E, A_4, A_5\}, \emptyset \rangle$ and DECONSTRUCT(B_2, G, π) returns $\langle \emptyset, \{D_1, D_2\}, \emptyset \rangle$. The lack of summations on line A5:3 of causes EXTRACT to iterate through the atomic expression contained in B_1 and B_2 directly on line A5:6, since neither of them have any sub-expressions of their own.

Only A_2 contains a sum at this point. The iteration over the terms of A_2 on line A5:10 finds that the only term that does not contain X is $P(Z_2)$ on line A5:11. Let us denote the atomic expression with the value $P(Z_2)$ as C_1 and the atomic expression resulting from the extraction as C_2 which now has the value

$$\sum_{X} P(Y|Z_2, X, Z_3, Z_1)P(Z_3|Z_2, X)P(X|Z_2).$$

This completes the extraction and results in an expression B'_1 such that

$$B'_1 = \langle \emptyset, \{C_1, C_2, E, A_4, A_5\}, \emptyset \rangle.$$

The expression B_2 remains unchanged.

q-simplify is now able to proceed. Neither B'_1 nor B_2 contain sub-expression so the loop on line A6:7 is not entered, and we are only subtracting their common atomic expressions in the loop on line A6:16. It is easy to see that $A_4 = D_1$ and $C_1 = D_2$, so they are removed from both B'_1 and B_2. Finally, the expressions corresponding to the numerator and denominator are returned.

To summarize, we began with the expression

$$P(Z_1|Z_2, X)P(Z_3|Z_2)\frac{\sum_X P(Y|Z_2, X, Z_3, Z_1)P(Z_3|Z_2, X)P(X|Z_2)P(Z_2)}{\sum_{X,Y} P(Y|Z_2, X, Z_3, Z_1)P(Z_3|Z_2, X)P(X|Z_2)P(Z_2)} \times \sum_{X,Z_3,Y} P(Y|Z_2, X, Z_3, Z_1)P(Z_3|Z_2, X)P(X|Z_2)P(Z_2).$$

and successfully simplified it into

$$P(Z_1|Z_2, X)P(Z_2)\sum_{X} P(Y|Z_2, X, Z_3, Z_1)P(Z_3|Z_2, X)P(X|Z_2).$$

6. Discussion

We have presented a formal definition of topologically consistent atomic expressions and simplification sets and provided a sound and complete algorithm to find these sets for a given expression. We also discussed some general techniques that apply to a more general class of these expressions. Algorithm 7 and Algorithm 8, presented in Appendix A, have been previously implemented in the R package causaleffect (Tikka and Karvanen, 2017a). We have updated the package to include all of the simplification procedures presented in this paper and they can be applied to all causal effect and conditional causal effect expressions derived from identification procedures. Our definition of topologically consistent atomic expressions is similar to g-functionals that can be used to characterize identifiability results under special conditions (Shpitser and Tchetgen Tchetgen, 2016).

It is plausible that these procedures could also be extended into other causal inference results, such as formulas for z-identifiability, transportability and meta-transportability of causal effects. The extensions are non-trivial however, since transportability formulas contain terms with distributions from multiple domains and z-identifiable causal effects contain do-operators in the conditioning sets which would require the implementation of the rules of do-calculus into Algorithm 1. Do-calculus consists of three inference rules that can

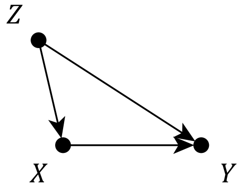

Figure 4: A graph G for a situation where simplification fails

be used to manipulate probabilities involving the do-operator (Pearl, 2009). Currently, we operate only on expressions that do not involve the do-operator. In fact, in our procedure it is not required to know the original causal query that produced the result.

Simpler expressions have many useful properties. They can help in understanding and communicating results and evaluating them saves computational resources. Estimation accuracy can also be improved in some cases when variables that are present in the original expression suffer from missing data or measurement error. One example where the benefits of simplification are realized can be found in (Hyttinen et al., 2015), where expressions of causal effects are derived and repeatedly evaluated for a large number of causal models.

Our approach to simplification stems from the nature of causal effect expressions. In our setting, a question still remains whether simplification sets completely characterize all situations where a variable can be eliminated from an atomic expression. One might also consider simplification in a general setting, where we do not assume topological consistency or any other constraints for the atomic expressions. In this case a 'black box' definition for simplification could be considered, where we simply require that when the sum over a variable of interest is completed we are again left with another atomic expression without this variable in the summation set. This framework is theoretically interesting but we are not aware of any potential applications.

The worst case time complexity of Algorithm 1 is difficult to gauge and is a topic for further research. One can observe that the performance of the algorithm is highly dependent on the size of the differences of the conditioning sets between adjacent terms. Both Algorithm 2 and Algorithm 3 iterate through the subsets of these differences and check d-separation criteria for each subset. Thus dynamic programming solutions could be implemented to further improve performance by collecting the results of these checks. Previously determined conditional independences would not need to be checked again and could be retrieved from memory instead.

In some cases, simplification has some apparent connections to identifiability. Consider the graph G of Figure 4. In this graph the causal effect of X on Y is identifiable, and its expression is

$$\sum_{Z} P(Y|Z,X)P(Z).$$

If we let Z be an unobserved variable instead, then G depicts the well-known bow-arc graph, where the same causal effect is unidentifiable. This corresponds to an unsuccessful attempt to remove Z from the expression of the causal effect. However, we cannot know beforehand whether an expression for a causal effect is going to be atomic or not, so we cannot use our algorithm to derive identifiability in general.

A reviewer suggested a simplification algorithm where the ID algorithm would be applied to latent projections (Pearl and Verma, 1991) onto the variables to be marginalized. This algorithm would be able to solve many, but not all, simplification tasks. Importantly, in the example presented in Figure 1, we cannot make any variables latent, as we are interested in the causal effect of X on all of the other variables. A reviewer also suggested that simplified expressions could be categorized into those that are obtained through latent projections and those that are not. This categorization might give additional insight into the topic.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback that greatly helped us to improve this paper. We also thank Lasse Leskelä for his comments.

Appendix A. Topological Consistency of Causal Effect Formulas

We prove the statement that every causal effect formula returned by the algorithms of Shpitser and Pearl (2006a,b) can be represented by using products and quotients of π-consistent atomic expressions such that π is a topological ordering of G.

We use the notation $G[\mathbf{X}]$ to denote an induced subgraph, which is obtained from G by removing all vertices not in $\mathbf{X}$ and by keeping all edges between the vertices of $\mathbf{X}$ in G. Here $G_{\overline{\mathbf{X}},\underline{\mathbf{Z}}}$ means the graph that is obtained from G by removing all incoming edges of $\mathbf{X}$ and all outgoing edges of $\mathbf{Z}$. We say that G is an *I-map* of P if P admits the causal Markov factorization with respect to $G = \langle \mathbf{V}, \mathbf{E} \rangle$, which is

$$P = \prod_{i=1}^{n} P(V_i | Pa^*(V_i)_G) \prod_{j=1}^{k} P(U_j),$$

where $Pa^*(\cdot)$ contains unobserved parents as well.

Consider first lines 2, 3, 4 and 7 of Algorithm 7 where recursive calls occur and let π_r be the topological ordering of the graph in the previous recursion step. Line 2 limits the identification procedure to the ancestors of $\mathbf{Y}$ so we can still obtain an expression that topologically consistent with respect to a topological ordering obtained from π_r by removing non-ancestors. Lines 3 and 4 make no changes to the distribution P and the graph G. On line 7 the induced subgraph $G[\mathbf{S}']$ in the next call is a C-component, but the joint distribution in this case is a π_r-consistent expression

$$P(\mathbf{S}') = \prod_{V_i \in \mathbf{S}'} P(V_i | V_i^{\pi_r} \cap \mathbf{S}', v_i^{\pi_r} \setminus \mathbf{s}'),$$

since every conditioning set is of the form $V_i^{\pi_r}$ when we only consider variables instead of their values, so we obtain

$$P(\mathbf{S}') = \prod_{V_i \in \mathbf{S}'} P(V_i | V_i^{\pi_r}),$$

Furthermore, any expression returned from line 7 will now be π_r-consistent. Thus all recursive calls retain topological consistency with respect to π.

Consider now the non-recursive terminating calls on lines 1 and 6. Consider line 1 first. If line two was triggered previously, we can factorize $P(\mathbf{V})$ in such a way that each variable is conditioned by its ancestors, since the ancestors of ancestors of $\mathbf{Y}$ are by definition ancestors of $\mathbf{Y}$. If line 7 was triggered previously we already know that the joint distribution was previously factorized in a π_r-consistent fashion. If line 3 or 4 was triggered previously, we know that they have not imposed any changes on P of G. Line 6 clearly produces a π_r consistent end result. Lines 4, 6 and 7 can only produce either products of quotients. By noting that π_r-consistency implies π-consistency, we have that the result of the algorithm can always be represented by using products and quotients of π-consistent atomic expressions.

Algorithm 7 The causal effect of intervention $do(\mathbf{X} = \mathbf{x})$ on $\mathbf{Y}$ (Shpitser and Pearl, 2006a).

INPUT: Value assignments $\mathbf{x}$ and $\mathbf{y}$, joint distribution $P(\mathbf{v})$ and a DAG $G = \langle \mathbf{V}, \mathbf{E} \rangle$. G is an I-map of P.

OUTPUT: Expression for $P_{\mathbf{x}}(\mathbf{y})$ in terms of $P(\mathbf{v})$ or $\mathbf{FAIL}(F, F')$.

```
   function ID($\mathbf{y}, \mathbf{x}, P, G$)
1:  if $\mathbf{x} = \emptyset$, then
       return $\sum_{v \in \mathbf{v} \setminus \mathbf{y}} P(\mathbf{v})$.
2:  if $\mathbf{V} \neq An(\mathbf{Y})_G$, then
       return ID($\mathbf{y}, \mathbf{x} \cap An(\mathbf{Y})_G, P(An(\mathbf{Y})_G), G[An(\mathbf{Y})_G]$).
3:  Let $\mathbf{W} = (\mathbf{V} \setminus \mathbf{X}) \setminus An(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$.
    if $\mathbf{W} \neq \emptyset$, then
       return ID($\mathbf{y}, \mathbf{x} \cup \mathbf{w}, P, G$).
4:  if $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}_1], \ldots, G[\mathbf{S}_k]\}$, then
       return $\sum_{v \in \mathbf{v} \setminus (\mathbf{y} \cup \mathbf{x})} \prod_{i=1}^{k}$ ID($\mathbf{s}_i, \mathbf{v} \setminus \mathbf{s}_i, P, G$).
    if $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}]\}$, then
5:     if $C(G) = \{G\}$, then
          throw FAIL($G, G[\mathbf{S}]$).
6:     if $G[\mathbf{S}] \in C(G)$, then
          return $\sum_{v \in \mathbf{s} \setminus \mathbf{y}} \prod_{V_i \in \mathbf{S}} P(v_i | v_i^{\pi})$.
7:     if $(\exists \mathbf{S}')\mathbf{S} \subset \mathbf{S}'$ such that $G[\mathbf{S}'] \in C(G)$, then
          return ID($\mathbf{y}, \mathbf{x} \cap \mathbf{s}', \prod_{V_i \in \mathbf{S}'} P(V_i | V_i^{\pi} \cap \mathbf{S}', v_i^{\pi} \setminus \mathbf{s}'), G[\mathbf{S}']$).
```

The claim is now apparent for Algorithm 8 since line 2 is eventually called for every conditional causal effect.

Algorithm 8 The causal effect of intervention $do(\mathbf{X} = \mathbf{x})$ on $\mathbf{Y}$ given $\mathbf{Z}$ (Shpitser and Pearl, 2006b).

INPUT: Value assignments $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$, joint distribution $P(\mathbf{v})$ and a DAG $G = \langle \mathbf{V}, \mathbf{E} \rangle$. G is an I-map of P.

OUTPUT: Expression for $P_{\mathbf{x}}(\mathbf{y}|\mathbf{z})$ in terms of $P(\mathbf{v})$ or $\mathbf{FAIL}(F, F')$.

```
function IDC($\mathbf{y}, \mathbf{x}, \mathbf{z}, P, G$)
if $\exists Z \in \mathbf{Z}$ such that $(\mathbf{Y} \perp\!\!\!\perp Z | \mathbf{X}, \mathbf{Z} \setminus \{Z\})_{G_{\overline{\mathbf{X}},\underline{\mathbf{Z}}}}$ then
    return IDC($\mathbf{y}, \mathbf{x} \cup \{z\}, \mathbf{z} \setminus \{z\}, P, G$).
else let $P' = \mathbf{ID}(\mathbf{y} \cup \mathbf{z}, \mathbf{x}, P, G)$.
return $P' / \sum_{y \in \mathbf{y}} P'$
```

Appendix B. Proof of Theorem 6

Proof By direct calculation we obtain

$$\begin{aligned}
P_A &= \sum_{V_j} \prod_{i=1}^{n} P(V_i|\mathbf{C}_i) \\
&= \prod_{V_i < V_j} P(V_i|\mathbf{C}_i) \sum_{V_j} \prod_{V_i \geq V_j} P(V_i|\mathbf{C}_i) \\
&= \prod_{V_i < V_j} P(V_i|\mathbf{C}_i) \sum_{V_j} \frac{P(V_{\pi(p)}, \ldots, V_{\pi(q)}|\mathbf{D})}{\prod_{U \in \mathbf{M}} P(U|\mathbf{E}_U)} \\
&= \prod_{V_i < V_j} P(V_i|\mathbf{C}_i) \frac{P(V_{\pi(p+1)}, \ldots, V_{\pi(q)}|\mathbf{D})}{\prod_{U \in \mathbf{M}} P(U|\mathbf{E}_U)} \\
&= \prod_{V_i < V_j} P(V_i|\mathbf{C}_i) \prod_{V_i > V_j} P(V_i|\mathbf{D}_i) := P_{A'},
\end{aligned}$$

where the sets $\mathbf{D}_i$ are obtained from the factorization of the joint term such that A' is a π^*-consistent where π^* is obtained from π by removing V_j from the ordering. To justify the equalities, we first note that terms of variables $V_i < V_j$ do not contain V_j and can be brought outside the sum.

To obtain the third equality, we multiply by $[\prod_{U \in \mathbf{M}} P(U|\mathbf{E}_U)]/[\prod_{U \in \mathbf{M}} P(U|\mathbf{E}_U)]$ and apply condition (2) of Definition 5 on the right-hand side as licensed by condition (3) of the definition. To obtain the fourth equality, we simply carry out the summation in terms of V_j. Conditions (2) and (3) of Definition 5 make it possible to refactorize the joint term into product terms so that the terms corresponding to variables $U \in \mathbf{M}$ remain unchanged and can be divided out once more. Thus we obtain the last equality, and an expression that no longer contains V_j and has the same value as A. ■

Appendix C. Derivation of the Causal Effect in the Introductory Example

We present the derivation of the causal effect of X on Y, Z_3, Z_2, Z_1 in the graph G of Figure 1 using Algorithm 7. We fix topological ordering of G as $Z_2 < X < Z_1 < Z_3 < Y$. The original call $\text{ID}(\{Y, Z_1, Z_2, Z_3\}, \{X\}, P(\mathbf{V}), G)$ fires line 4 and results in three new recursive calls. We have

$$P_X(Y, Z_3, Z_1, Z_2) = P_{Y,Z_3,X,Z_2}(Z_1)P_{Y,Z_1,X,Z_2}(Z_3)P_{Z_3,Z_1,X}(Y, Z_2), \tag{5}$$

as the graph $G[\mathbf{V} \setminus \{X\}]$ has three C-components formed by the sets $\{Z_1\}$, $\{Z_3\}$ and $\{Y, Z_2\}$, respectively.

The first recursive call $\text{ID}(\{Z_1\}, \{Y, Z_3, X, Z_2\}, P(\mathbf{V}), G)$ fires line 2 because Z_3 and Y are not ancestors of Z_1. The next call $\text{ID}(\{Z_1\}, \{X, Z_2\}, P(Z_1, X, Z_2), G[\{Z_1, X, Z_2\}])$ fires line 6 because $C(G[\{Z_1\}])$ contains only one C-component and it is not part of a larger C-component in the graph of the current recursion stage. We have

$$P_{Y,Z_3,X,Z_2}(Z_1) = P_{X,Z_2}(Z_1) = P(Z_1|X, Z_2). \tag{6}$$

To obtain $P_{Y,Z_1,X,Z_2}(Z_3)$ we call $\text{ID}(\{Z_3\}, \{Y, Z_1, X, Z_2\}, P(\mathbf{V}), G)$ which also fires line 2 because X, Z_1 and Y and not ancestors of Z_3. Calling $\text{ID}(\{Z_3\}, \{Z_2\}, P(Z_3, Z_2), G[\{Z_3, Z_2\}])$ fires line 6 $C(G[\{Z_3\}])$ contains only one C-component and it is not part of a larger C-component in the graph of the current recursion stage. We have

$$P_{Y,Z_1,X,Z_2}(Z_3) = P_{Z_2}(Z_3) = P(Z_3|Z_2). \tag{7}$$

To obtain the last term we call $\text{ID}(\{Y, Z_2\}, \{Z_3, Z_1, X\}, P(\mathbf{V}), G)$. The subgraph $G[\mathbf{V} \setminus \{Z_3, Z_1, X\}] = G[\{Y, Z_2\}]$ has only one C-component, but it is part of a larger C-component formed by the set $\mathbf{S}' = \{Y, Z_3, X, Z_2\}$ in the current graph G. Line 7 is fired resulting in

$$\text{ID}(\{Y, Z_2\}, \{Z_3, X\}, P(Y|Z_3, Z_1, X, Z_2)P(Z_3|Z_2, X)P(X|Z_2)P(Z_2), G[\mathbf{S}']). \tag{8}$$

This call fires line 2 since X is not an ancestor of Y in the graph $G[\mathbf{S}']$. Letting $\mathbf{T} = \mathbf{S}' \setminus \{X\} = \{Y, Z_3, Z_2\}$ the next call is

$$\text{ID}(\{Y, Z_2\}, \{Z_3\}, \sum_X P(Y|Z_3, Z_1, X, Z_2)P(Z_3|Z_2, X)P(X|Z_2)P(Z_2), G[\mathbf{T}]). \tag{9}$$

This time we trigger line 6 because $G[\mathbf{T} \setminus \{Z_3\}]$ has only one C-component and there is no larger C-component of $G[\mathbf{T}]$ that would contain it. We obtain

$$\begin{aligned} P_{Z_3,Z_1,X}(Y, Z_2) &= P_{Z_3,Z_1,X}(Y, Z_2) \\ &= P_{Z_3,X}(Y, Z_2) \\ &= P_{Z_3}(Y, Z_2) \\ &= P^*(Y|Z_3, Z_2)P^*(Z_2), \end{aligned} \tag{10}$$

where P^* is the distribution of the current recursion stage, that is

$$P^*(Y, Z_3, Z_2) = \sum_X P(Y|Z_3, Z_1, X, Z_2)P(Z_3|Z_2, X)P(X|Z_2)P(Z_2).$$

In order to represent the conditional probability on the last line of (10), we write

$$
\begin{aligned}
P^*(Y|Z_3,Z_2)P^*(Z_2) &= \frac{P^*(Y,Z_3,Z_2)}{P^*(Z_3,Z_2)}P^*(Z_2) \\
&= \frac{P^*(Y,Z_3,Z_2)}{\sum_Y P^*(Y,Z_3,Z_2)}\sum_{Y,Z_3} P^*(Y,Z_3,Z_2) \\
&= \frac{\sum_X P(Y|Z_2,X,Z_3,Z_1)P(Z_3|Z_2,X)P(X|Z_2)P(Z_2)}{\sum_{X,Y} P(Y|Z_2,X,Z_3,Z_1)P(Z_3|Z_2,X)P(X|Z_2)P(Z_2)}\times \\
&\sum_{X,Z_3,Y} P(Y|Z_2,X,Z_3,Z_1)P(Z_3|Z_2,X)P(X|Z_2)P(Z_2).
\end{aligned} \tag{11}
$$

Finally, we gather the results of our subproblems in (6), (7) and (11), and insert them back into the equation in (5) which yields

$$
\begin{aligned}
P_X(Y,Z_3,Z_1,Z_2) = P(Z_1|Z_2,X)P(Z_3|Z_2)\times \\
\frac{\sum_X P(Y|Z_2,X,Z_3,Z_1)P(Z_3|Z_2,X)P(X|Z_2)P(Z_2)}{\sum_{X,Y} P(Y|Z_2,X,Z_3,Z_1)P(Z_3|Z_2,X)P(X|Z_2)P(Z_2)}\times \\
\sum_{X,Z_3,Y} P(Y|Z_2,X,Z_3,Z_1)P(Z_3|Z_2,X)P(X|Z_2)P(Z_2)
\end{aligned}
$$

as the formula for the causal effect.

Appendix D. Proof of Theorem 7

Proof (i) Suppose that simplify(A,G,π) has returned an expression with variable V_j eliminated. Because the computation completed successfully, we have that each application of join and insert succeed. We can rewrite the value of A as

$$\prod_{V_i<V_j} P(V_i|\mathbf{C}_i)\sum_{V_j}\prod_{V_i\geq V_j} P(V_i|\mathbf{C}_i),$$

where the terms $P(V_i|\mathbf{C}_i)$ such that $V_i < V_j$ can be brought outside the sum over V_j, because they cannot contain V_j. The functions join and insert use only standard rules of probability calculus, which can be seen on line 10 of Algorithm 2 and line 7 of Algorithm 3, and thus every new formation of a joint distribution $P(\mathbf{J}|\mathbf{D})$ has been valid. Once again we rewrite the value of A as

$$\prod_{V_i<V_j} P(V_i|\mathbf{C}_i)\sum_{V_j} P(\mathbf{J}|\mathbf{D}),$$

which means that condition (2) of Definition 5 is now satisfied, as we have obtained a joint term from the original product terms.Because $V_j \in \mathbf{J}$ we can carry out the summation which yields

$$\prod_{V_i<V_j} P(V_i|\mathbf{C}_i)\cdot P(\mathbf{J}\setminus\{V_j\}|\mathbf{D}),$$

Because Algorithm 1 succeeds, we know that every insertion is canceled out by factorize. To complete the procedure we obtain a new factorization without V_j resulting in an atomic expression A' that no longer contains V_j. Condition (3) of Definition 5 is satisfied by the definition of insert, because the function always checks the conditional independence with the current summation variable on line 7. Both conditions for simplification sets have been satisfied by construction.

(ii) Suppose that there exists a collection of simplification sets of A with respect to V_j. For the sake of clarity, assume further that $V_n = V_j$. This assumption lets us only consider those terms that are relevant to the simplification of V_j, as we can always move conditionally independent terms outside the summation and consider only the expression remaining inside the sum. Let us first assume that $\mathbf{M} = \emptyset$. In this case condition (2) simply reads

$$\prod_{V_i \geq V_j} P(V_i|\mathbf{C}_i) = P(V_j, \ldots, V_1|\mathbf{D}),$$

and that the product terms are a factorization of the joint term. However, we want to show that they also provide a factorization that agrees with the topological ordering. Because A is $\pi-$consistent, for any two variables $V > W$ we have that $\mathbf{C}_W \subseteq V^\pi$ which enables us to consider the summations from V_k up to V_1 for $k = 1, \ldots, j-1$, which results in

$$\sum_{V_k,\ldots,V_1} \prod_{V_i \geq V_j} P(V_i|\mathbf{C}_i) = \sum_{V_k,\ldots,V_1} P(V_j, \ldots, V_1|\mathbf{D}) = P(V_j, \ldots, V_{k+1}|\mathbf{D}).$$

We obtain for $k = j-1, \ldots, 1$

$$\begin{aligned} P(V_j|\mathbf{C}_j) &= P(V_j|\mathbf{D}) \\ P(V_j|\mathbf{C}_j)P(V_{j-1}|\mathbf{C}_{j-1}) &= P(V_j, V_{j-1}|\mathbf{D}) \\ &\vdots \\ P(V_j|\mathbf{C}_j) \cdots P(V_2|\mathbf{C}_2) &= P(V_j, \ldots, V_2|\mathbf{D}) \\ P(V_j|\mathbf{C}_j) \cdots P(V_2|\mathbf{C}_2)P(V_1|\mathbf{C}_1) &= P(V_j, \ldots, V_1|\mathbf{D}). \end{aligned} \tag{12}$$

From the last and second to last equation we can obtain

$$P(V_j, \ldots, V_2|\mathbf{D})P(V_1|\mathbf{C}_1) = P(V_j, \ldots, V_1|\mathbf{D}),$$

and by dividing with the first term from the left hand side we obtain

$$P(V_1|\mathbf{C}_1) = P(V_1|V_j, \ldots, V_2, \mathbf{D}).$$

In fact, we can do this for any two subsequent equations in (12) to obtain

$$P(V_i|\mathbf{C}_i) = P(V_i|V_j, \ldots, V_{i+1}, \mathbf{D}), \quad i = 1, \ldots, j-1$$

Algorithm 1 operates by starting from V_1, so we still have to show it succeeds in constructing the joint term. Using the previous results we can rewrite the original equation as

$$\prod_{V_i \geq V_j} P(V_i|\mathbf{C}_i) = \prod_{V_i \geq V_j} P(V_i|\mathbf{C}_i^*),$$

where $\mathbf{C}_i^* = \mathbf{D} \cup \{V_j, \ldots, V_{i+1}\}$ for $i < j$ and $\mathbf{C}_j^* = \mathbf{D}$. From this we obtain

$$\begin{aligned} P(V_1|\mathbf{C}_1) &= P(V_1|\mathbf{C}_1^*) \\ P(V_1|\mathbf{C}_1^*)P(V_2|\mathbf{C}_2) &= P(V_1, V_2|\mathbf{C}_2^*) \\ &\vdots \\ P(V_1, \ldots, V_{j-1}|\mathbf{C}_{j-1}^*)P(V_j|\mathbf{C}_j) &= P(V_j, \ldots, V_1|\mathbf{C}_j^*). \end{aligned} \tag{13}$$

The function join will succeed every time since the for-loop starting on line 7 of Algorithm 2 will discover the conditional independence properties allowing the previous equalities in (13) to take place. Thus Algorithm 1 will return an atomic expression with the variable V_j eliminated from the summation set.

Assume now that $\mathbf{M} \neq \emptyset$ and let $\mathbf{V} = V[A]$ and. In this case condition (2) allows us to write

$$\prod_{U \in \mathbf{M}} P(U|\mathbf{E}_U) \prod_{V_i \geq V_j} P(V_i|\mathbf{C}_i) = P(\mathbf{V}, \mathbf{M}|\mathbf{D}),$$

and furthermore, we have that these product terms are a factorization of the joint term. First, we aim to reduce the number of variables in $\mathbf{M}$ to be considered. This is done because Algorithm 1 always starts and finishes the construction of the joint term with a variable in $\mathbf{V}$. We categorize each $U \in \mathbf{M}$ into three disjoint sets. We define

$$\mathbf{M}^- := \{U \in \mathbf{M} \mid U \notin \bigcup_{k=1}^{j} \mathbf{C}_k\}, \mathbf{M}^+ := \{U \in \mathbf{M} \mid U \in \bigcap_{k=1}^{j} \mathbf{C}_k\} \text{ and}$$

$$\mathbf{M}^* := \mathbf{M} \setminus (\mathbf{M}^- \cup \mathbf{M}^+).$$

First, we show that we can ignore variables in $\mathbf{M}^-$ by obtaining a new factorization without them. It follows from the definition of $\mathbf{M}^-$ and (2) that we can compute the marginalization as follows

$$\begin{aligned} P(\mathbf{V}, \mathbf{M} \setminus \mathbf{M}^-|\mathbf{D}) &= \sum_{U \in \mathbf{M}^-} P(\mathbf{V}, \mathbf{M}|\mathbf{D}) \\ &= \sum_{U \in \mathbf{M}^-} \prod_{U \in \mathbf{M}} P(U|\mathbf{E}_U) \prod_{V_i \geq V_j} P(V_i|\mathbf{C}_i) \\ &= \prod_{V_i \geq V_j} P(V_i|\mathbf{C}_i) \sum_{U \in \mathbf{M}^-} \prod_{U \in \mathbf{M}} P(U|\mathbf{E}_U) \\ &= \prod_{U \in \mathbf{M} \setminus \mathbf{M}^-} P(U|\mathbf{E}_U) \prod_{V_i \geq V_j} P(V_i|\mathbf{C}_i). \end{aligned}$$

We have a new factorization without any variables in $\mathbf{M}^-$. Similarly, we can eliminate the variables in $\mathbf{M}^+$ from our factorization. It follows from the definition of $\mathbf{M}^+$ that for all $U \in \mathbf{M}^+$ we have that $\mathbf{E}_U \subseteq \mathbf{D}$. From this we obtain

$$\prod_{U \in \mathbf{M}^+} P(U|\mathbf{E}_U) = P(\mathbf{M}^+|\mathbf{D}).$$

We can now write

$$\begin{aligned}
P(\mathbf{V},\mathbf{M}^*|\mathbf{D},\mathbf{M}^+) &= \frac{P(\mathbf{V},\mathbf{M}\setminus\mathbf{M}^-)}{P(\mathbf{M}^+|\mathbf{D})} \\
&= \frac{\prod_{U\in\mathbf{M}\setminus\mathbf{M}^-} P(U|\mathbf{E}_U)\prod_{V_i\geq V_j} P(V_i|\mathbf{C}_i)}{\prod_{U\in\mathbf{M}^+} P(U|\mathbf{E}_U)} \\
&= \prod_{U\in\mathbf{M}^*} P(U|\mathbf{E}_U)\prod_{V_i\geq V_j} P(V_i|\mathbf{C}_i).
\end{aligned}$$

Thus it suffices to consider the factorization given by

$$\prod_{U\in\mathbf{M}^*} P(U|\mathbf{E}_U)\prod_{V_i\geq V_j} P(V_i|\mathbf{C}_i) = P(\mathbf{V},\mathbf{M}^*|\mathbf{D}^*), \tag{14}$$

where $\mathbf{D}^* = \mathbf{D}\cup\mathbf{M}^+$.

Next, we will order the variables in $\mathbf{M}^*$. For each $U\in\mathbf{M}^*$ we find the largest index $u\in\{1,\ldots,j-1\}$ such that $U\in\mathbf{C}_u$. This choice is well defined, since by definition at least one such index exists. Furthermore, as the product terms in (14) are a factorization of the joint term, the conditioning sets are increasing and we have that $U\notin\mathbf{C}_i$ for all $i\geq u+1$. In the case that multiple variables $U_i\in\mathbf{M}^*$ for some set of indices $i\in\mathbf{I}$ share the same index u, we may redefine M^* such that $U_i, i\in\mathbf{I}$ are replaced by a single variable U_I such that $\prod_{i\in I} P(U_i|\mathbf{E}_{U_i}) = P(U_I|\mathbf{E}_{U_I})$, where $\mathbf{E}_{U_I} = \cap_{i\in\mathbf{I}}\mathbf{E}_{U_i}$. Thus we can assume that for any two variables $U_1, U_2\in\mathbf{M}^*$ we have that $u_1\neq u_2$. We can now order the variables in $\mathbf{M}^*$ by their respective indices u such that $U_1 > U_2 > \ldots > U_m$ and $u_1 < u_2 < \ldots < u_m$.

Nest we will extend the ordering to include all of the variables in the set $\mathbf{V}$. We let $\mathbf{Q} := \mathbf{V}\cup\mathbf{M}^*$ and find an ordering of this set such that it agrees with induced ordering ω of the variables in $\mathbf{V}$ and with the ordering of the indices $u_1,\ldots,u_m$. A new factorization given by this ordering can be defined as follows:

$$Q_k = \begin{cases} V_{k-m} & k > u_m, \\ V_{k-l} & u_l < k < u_{l+1}, \\ V_k & k < u_1, \\ U_l & k = u_l. \end{cases} \qquad \mathbf{D}_k = \begin{cases} \mathbf{C}_{k-m} & k > u_m, \\ \mathbf{C}_{k-l} & u_l < k < u_{l+1}, \\ \mathbf{C}_k & k < u_1, \\ \mathbf{E}_{U_l} & k = u_l. \end{cases}$$

We can now rewrite the factorization of (14) as

$$\prod_{k=1}^{n+m} P(Q_k|\mathbf{D}_k) = P(\mathbf{Q}|\mathbf{D}^*), \tag{15}$$

We can now apply the same procedures as in the case of $\mathbf{M}=\emptyset$ with the exception that insert succeeds where join fails with terms containing Q_k and Q_{k+1} when $k = l-1$ for all $l = 1,\ldots,m$. The success of insert is guaranteed by condition (3), as the function will find this conditional independence on line 10 of Algorithm 3. Also, factorize will remove all additional terms that were introduced in the process, which is made possible by condition (3) and the definition of the factorization of $P(\mathbf{Q}|\mathbf{D}^*)$. After the summation over V_j is carried out, the conditional independence between V_j and the variables $U \in \mathbf{M}^*$ ensures that their respective terms are equal to the original factorization before the summation was carried out when the new factorization is constructed so that it agrees with the ordering of the set $\mathbf{Q}$. Thus an atomic expression is returned with the variable V_j eliminated with the same value as the original atomic expression. ■

References

D. H. Bailey, Borwein J. M., and D. A. Kaiser. Automated simplification of large symbolic expressions. *Journal of Symbolic Computation*, 60:120–136, 2014.

E. Bareinboim and J. Pearl. Causal inference by surrogate experiments: z-identifiability. In N. de Freitas and K. Murphy, editors, *Proceedings of the Twenty-Eight Conference on Uncertainty in Artificial Intelligence*, pages 113–120. AUAI Press, 2012.

E. Bareinboim and J. Pearl. Meta-transportability of causal effects: a formal approach. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*, pages 135–143, 2013a.

E. Bareinboim and J. Pearl. A general algorithm for deciding transportability of experimental results. *Journal of Causal Inference*, 1:107–134, 2013b.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

J. Carette. Understanding expression simplification. In *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation*, pages 72–79, New York, 2004. ACM.

D. Geiger, T. Verma, and J. Pearl. Identifying independence in Bayesian networks. *Networks*, 20(5):507–534, 1990.

Y. Huang and M. Valtorta. Pearl's calculus of intervention is complete. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, pages 217–224. AUAI Press, 2006.

A. Hyttinen, F. Eberhardt, and M. Järvisalo. Do-calculus when the true graph is unknown. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, pages 395–404. AUAI Press, 2015.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B (Methodological)*, 50(2):157–224, 1988.

Maxima. Maxima, a computer algebra system. version 5.34.1, 2014. URL `http://maxima.sourceforge.net/`.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, 1988.

J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, New York, 2nd edition, 2009.

J. Pearl and T. S. Verma. A theory of inferred causation. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 441–452, 1991.

I. Shpitser and J. Pearl. Identification of joint interventional distributions in recursive semi-Markovian causal models. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, pages 1219–1226. AAAI Press, 2006a.

I. Shpitser and J. Pearl. Identification of conditional interventional distributions. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, pages 437–444. AUAI Press, 2006b.

I. Shpitser and E. Tchetgen Tchetgen. Causal inference with a graphical hierarchy of interventions. *Annals of Statistics*, 44(6):2433–2466, 2016.

I. Shpitser, T. S. Richardson, and J. M. Robins. An efficient algorithm for computing interventional distributions in latent variable causal models. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 661–670. AUAI Press, 2011.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search.* MIT press, 2nd edition, 2000.

S. Tikka and J. Karvanen. Identifying causal effects with the R package causaleffect. *Journal of Statistical Software*, 76(12):1–30, 2017a.

S. Tikka and J. Karvanen. Simplifying probabilistic expressions in causal inference. *Journal of Machine Learning Research*, 18(36):1–30, 2017b. URL `http://jmlr.org/papers/v18/16-166.html`.

Wolfram Research Inc. Mathematica, version 10.3, 2015.