UNIVERSITY OF JYVÄSKYLÄ
DEPARTMENT OF MATHEMATICS
AND STATISTICS

REPORT 168

UNIVERSITÄT JYVÄSKYLÄ
INSTITUT FÜR MATHEMATIK
UND STATISTIK

BERICHT 168

IMPROVING IDENTIFICATION ALGORITHMS IN CAUSAL INFERENCE

SANTTU TIKKA

JYVÄSKYLÄ
2018

UNIVERSITY OF JYVÄSKYLÄ
DEPARTMENT OF MATHEMATICS
AND STATISTICS

REPORT 168

UNIVERSITÄT JYVÄSKYLÄ
INSTITUT FÜR MATHEMATIK
UND STATISTIK

BERICHT 168

IMPROVING IDENTIFICATION ALGORITHMS IN CAUSAL INFERENCE

SANTTU TIKKA

To be presented, with permission of the Faculty of Mathematics and Science
of the University of Jyväskylä, for public criticism in Auditorium H320
on August 24th, 2018, at 12 o'clock noon.

JYVÄSKYLÄ
2018

Editor: Pekka Koskela
Department of Mathematics and Statistics
P.O. Box 35 (MaD)
FI–40014 University of Jyväskylä
Finland

ISBN 978-951-39-7518-0 (print)
ISBN 978-951-39-7519-7 (pdf)
ISSN 1457-8905

University Printing House
Jyväskylä 2018

Abstract

Causal models provide a formal approach to the study of causality. One of the most useful features of causal modeling is that it enables one to make causal claims about a phenomenon using observational data alone under suitable conditions. This feature enables the analysis of interventions that may be infeasible to conduct in the real world for practical or ethical reasons. The uncertainty associated with the variables of interest is taken into account by including a probability distribution in the causal model, making it is possible to study the effects of external interventions by examining how this distribution is changed by the action. The probability distribution of a specific variable in a causal model perturbed by an outside intervention is the causal effect of that intervention on the variable.

One of the most fundamental problems of causal inference is determining whether a causal effect can be uniquely expressed in terms of the joint probability distribution over the observed variables in a given causal model. Causal effects that can be expressed in this way are called identifiable and they serve as the link between observational and experimental information. Complete solutions to the identifiability problem take the form of an algorithm that produces an expression in terms of observed quantities whenever the causal effect given as input is identifiable. However, completeness in this context refers only to the correctness and exhaustiveness of the methods. The formulas obtained as output from identifiability algorithms are often impractical and unnecessarily complicated.

The thesis augments the pre-existing identifiability methodology by providing a simplification procedure that drastically improves the complicated outputs in many cases. Simplification also has practical benefits when statistical estimation is considered if variables affected by bias or missing data no longer appear in the simplified expression. The thesis also introduces a new method called pruning, which aims to eliminate variables that are unnecessary for the identification task from the causal model itself. Finally, a variety of identification algorithms are implemented more complicated settings, such as when data are available from multiple domains. The methods are provided through the R package “causaleffect”.

Tiivistelmä

Kausaalimallit tarjoavat formaalin lähestymistavan kausaalisuuden tutkimiseen. Kausaalimallin yksi hyödyllisimmistä ominaisuuksista on, että sen avulla on mahdollista tehdä johtopäätöksiä syy-seuraussuhteista pelkästään havainnoivan aineiston perusteella sopivien olosuhteiden vallitessa. Tämä ominaisuus mahdollistaa sellaisten kokeiden analysoinnin, joita ei käytännöllisistä tai eettisistä syistä ole mahdollista toteuttaa todellisuudessa. Kiinnostuksen kohteena oleviin muuttujiin liittyvä epävarmuus otetaan huomioon lisäämällä kausaalimalliin todennäköisyysjakauma, mikä mahdollistaa ulkopuolisten interventioiden vaikutusten tutkimisen mallin todennäköisyysjakaumassa tapahtuvien muutosten kautta. Tietyn muuttujan jakauma kausaalimallissa, johon on tehty interventio mallin ulkopuolelta, on intervention kausaalivaikutus muuttujaan.

Yksi kausaalipäättelyn perustavanlaatuisimmista kysymyksistä on määrittää, milloin annettu kausaalivaikutus voidaan ilmaista yksikäsitteisesti annetun kausaalimallin muuttujien yhteisjakauman avulla. Tällaisia kausaalivaikutuksia kutsutaan identifioituviksi ja ne yhdistävät kokeisiin ja havaintoihin perustuvan informaation toisiinsa. Täydelliset ratkaisut identifioituvuusongelmaan esitetään yleensä algoritmin avulla, joka tuottaa identifioituvalle kausaalivaikutukselle lausekkeen havaittujen suureiden avulla esitettynä. Täydellisyys viittaa tässä yhteydessä ainoastaan menetelmien korrektiin toimintaan ja kattavuuteen. Identifoituvuusalgoritmien tulosteena tuotetut lausekkeet ovat usein vaikeaselkoisia ja tarpeettoman monimutkaisia.

Tämä väitöskirja täydentää jo olemassa olevaa identifioituvuusmetodien joukkoa tarjoamalla sievennysmenetelmän, joka huomattavasti parantaa monimutkaisia tuloksia monissa tapauksissa. Sievennyksestä on myös käytännön hyötyä tilastollisessa estimoinnissa jos muuttujia, joihin liittyy valintaharhaa tai puuttuvaa tietoa, ei enää esiinny sievennetyssä lausekkeessa. Väitöskirjassa esitellään myös karsintamenetelmä, jonka tavoitteena on eliminoida identifioituvuuden kannalta epäoleellisia muuttujia kausaalimallista itsestään. Väitöskirjassa myös implementoidaan lukuisia identifioituvuusmenetelmiä erilaisiin tilanteisiin, esimerkiksi joissa aineistoja on saatavilla useista eri lähteistä. Nämä menetelmät tarjotaan R-paketin “causaleffect” kautta.

Acknowledgements

I am the most grateful to my supervisor, Professor Juha Karvanen, for encouraging me to pursue a PhD and for supporting me throughout my studies. Sincere thanks also to Professor Elias Bareinboim for inviting me to visit him at Purdue University and for the numerous insightful conversations. I truly think we managed to create something remarkable that hopefully will someday see the light of day. I am grateful to Professors Jin Tian and Thomas Richardson for their work as pre-examiners of this thesis. A special thanks goes to Professor Richardson for his extremely thorough review and extensive commentary. I thank Associate professor Jose Peña for agreeing to serve as the opponent. I also wish to thank Professor Jukka Nyblom for the many interesting discussions about causality and my work.

I wish to express my gratitude to the Department of Mathematics and Statistics at the University of Jyväskylä for the financial support at the beginning of my PhD journey and for the excellent work conditions that it has provided throughout my graduate studies. I am deeply grateful to Professor Leena Lindström for including me in the project "Human induced trans-generational stress tolerance and invasion success" of the Academy of Finland. I also acknowledge the Finnish Cultural Foundation for the funding. The work belongs to the thematic research area "Decision analytics utilizing causal models and multiobjective optimization" of University of Jyväskylä.

Finally, I give my warmest thanks to my family and friends for their continuous support. This thesis would never have been completed without you.

Jyväskylä, May 2018

Santtu Tikka

List of original publications

This thesis consists of an introductory part and publications listed below.

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

II Tikka, S. and Karvanen, J. Simplifying probabilistic expressions in causal inference. *Journal of Machine Learning Research*, 18(36): 1–30, 2017.

III Tikka, S. and Karvanen, J. Enhancing identification of causal effects by pruning. *Journal of Machine Learning Research*, 18(194): 1–23, 2018.

Author contribution

The author of this thesis is the main contributor and writer of Articles I, II and III. In addition, he has implemented the algorithms described in Article I in a software package, which he is the sole maintainer of. The author also derived the simplification method of Article II and showed it to be complete for a large class of probabilistic expressions. The pruning criteria and the pruning identifiability algorithm of Article III were formulated together with the co-author. The author was responsible for proving the correctness of aforementioned criteria. The author has also further extended the package of Article I by incorporating all of the methods described in Articles II and III. The research questions, algorithms and inference methods have been formulated together with the co-author. The author had the main responsibility writing and ensuring the correctness of the programming code throughout Articles I–III.

Contents

Chapter 1

Introduction

Causality permeates our everyday discourse and it is an essential component in our understanding of the natural world. An intuitive understanding of causality has been necessary for not only our survival, but for other species of the animal kingdom as well. One will surely meet a quick end without learning to recognize that certain sounds imply the presence of a dangerous predator or that a plant having a specific shape or color is inedible. It seems though that for humanity the capacity to reason about causes and effects extends further to seemingly disconnected events or networks of multiple events and leads us to ask what causality truly is. The underlying processes in our minds are quick to pick apart notions such as "wet grass causes rain" and "eating ice cream causes drowning". Consensus can be reached about even more involved claims, such as "smoking causes lung cancer". However, even with questions that may seem trivial it is important to consider what they really mean.

Without a formal approach to causality, our intuition is always at risk of leading us to incorrect conclusions, the consequences of which range from harmless to catastrophic. A comparison can be drawn with the concept of probability which almost everyone has an intuitive understanding about. Without a proper formal characterization it is possible to derive many seemingly paradoxical results, that turn out to be straightforward to resolve with a rigorous approach and proper understanding of the issue. Just as there have been many formal definitions of probability (Bayes, 1763; Laplace, 1812), causality has also received its fair share of attempts at formalization that more or less tend to agree with our intuitive understanding of causes and effects (Neyman, 1923; Granger, 1969; Lewis, 1973a; Rubin, 1974). In this thesis, we will focus on a formalization by Pearl (1995, 2009) that is based on directed graphs and probability theory.

In the natural world, the temporal order of events often dictates the inherent directionality of causation. An event cannot precede its cause, but we can recognize systems where cyclical or mutual causation occurs. We can find such systems in nature, such as the decomposition of organic matter being used as nutrition for new life or two objects in motion colliding and causing each other to stop or change direction. When addressing more complicated systems, it is beneficial to focus on the distinct causes of the components involved in the system instead

of the system as a whole. This leads us to the notion of modularity that we also consider an inherent feature of causal influence. When considering a given effect, we are not required to be aware of the entire state of the rest of the world. Having knowledge of all the direct causes of the effect is enough for making accurate claims about the effect. When designing an experiment or formulating a statistical model, we have to make a decision about the scope of our approach. The variables we decide to include in the model and the precision of our instruments that we use to measure them all play a role in how well the model is able to capture reality. The same challenge of deciding the appropriate level of granularity in our analysis applies to making causal inferences. We will never be able to include every possible observable direct cause in our model due to both practical reasons and quantum interactions at the most fundamental level.

Graphs allow us to incorporate these intuitive concepts of directionality and modularity under a mathematical framework. They also provide a visual aid that may help researchers and applied scientists to be more precise about their claims and to better communicate the processes involved in the analysis. Graphs have emerged as a prominent tool in many scientific fields including econometrics (Strotz and Wold, 1960), artificial intelligence (Larrañaga and Moral, 2011), biomechanics (Andreassen et al., 1987) and social sciences (Blalock, 1971). Vertices of the graph represent variables of interest and edges of the graph represent direct causal relationships between those variables. The value of each variable is then completely determined by the values of those variables that are its direct causes, also known as its parents in graphical terms. Graphs associated with this interpretation are often called causal diagrams (Pearl, 2009). Prevalent issues involving real-world data such as selection bias and missing data have led to a causal formulation of these problems further necessitating the need for causal diagrams that extend the original interpretation. Graphs used to solve these issues include special vertices that have been given a role distinct from other vertices such as the mechanism responsible for the selection preference (Cooper, 2000; Geneletti et al., 2009; Didelez et al., 2010; Bareinboim et al., 2014) or missingness (Karvanen, 2015; Shpitser et al., 2015; Thoemmes and Mohan, 2015).

We are often faced with uncertainty regarding our causal system of interest. First, we may know that a variable is a direct cause of another, but do not have enough information to fully specify the functional form of the causal influence. Second, we are limited in the scope of our model and can only partly observe the direct causes of effects. Using probability theory, we are able to formally take these sources of uncertainty into account. As we have the capability to collect data on the variables that we have observed, we can postulate a joint probability distribution over them. In order to be able to make use of this distribution, we consider specific vertices in our graph unobserved and imbue them with a probability distribution. The flow of information in the network now induces a probability distribution over the observable effects. Additionally, we can make inferences about the functional relationships through this distribution by fixing a set of values for the causes of interest and analyzing the conditional probability of the effect conditioned on the fixed values.

To fully understand the causal nature of a phenomenon, a causal analysis can be conducted in three distinct steps: discovery, identification and estimation. In causal discovery, we are

not readily provided with a causal model but instead it has to be constructed from relevant data that is available or that can be collected. Thus the analysis begins by searching for a causal model that depicts the causal relationships in the data with sufficient accuracy. This domain of research is known as causal discovery, but labels such as causal induction have also been used (Griffiths and Tenenbaum, 2009). A statistical dependence between two variables is easy to confirm, but making a claim about a causal connection is much stronger. Many causal structures are able to generate the same dependence structure and usually specialized background knowledge is required to rule out undesirable structures. A common phrase "correlation does not imply causation" is an important principle here, since an unobserved common cause between two variables may result in the observed association between them (Reichenbach, 1956). There are many factors that have to be considered in causal discovery, such as latent confounding (Robins and Wasserman, 1999), direction of causation (Peters et al., 2009) and presence of cycles (Hyttinen et al., 2012).

Not every inferential question can be answered in the presence of uncertainty even when the causal model is known. At the heart of causal inference lies the question of whether the information encoded in our causal model is sufficient to uniquely characterize a specific query. This problem is known as the identifiability problem and it has been widely studied in many aspects of causal inference. Complete solutions have been derived for a general non-parametric setting (Tian, 2002; Shpitser and Pearl, 2006b; Huang and Valtorta, 2006b) as well as for various extensions (Bareinboim and Pearl, 2012a, 2014). Identifiability has also been studied in settings where the missing data mechanism or preferential selection is included in the causal model (Shpitser et al., 2015; Bareinboim and Pearl, 2012b) and when the causal relationships are assumed to be linear (Wright, 1921; Angrist et al., 1996; van der Zander and Liskiewicz, 2016; Chen et al., 2017). Identification is the second step in a causal analysis and various identification problems can be distinguished according to the causal hierarchy as defined by Pearl (2009). This hierarchy is useful for classifying various questions based on the type of information that is required to answer them and how difficult they are to understand conceptually. The first level of inquiry is called associative, which is concerned with queries such as how likely it is that a person who smokes will develop lung cancer. Questions at the associative level do not invoke the causal information embedded in our model and can be answered using the observed probability distribution alone.

The second level, intervention, is the primary focus of this thesis and it is placed higher in the causal hierarchy because it involves the effects of actions on the causal model that change the functional relationships involved. Intervention in this context means an ideal manipulation where only the causal mechanism of the target of the intervention is manipulated and other mechanisms of the model are left intact. For instance, we could consider how the likelihood of lung cancer changes if a smoker stops smoking or a non-smoker decides to start smoking while keeping other habits unchanged. Observational information about lung cancer prevalence alone is not enough to answer these questions since it does not account for this change in smoking behavior. Purely associational questions can be formulated also at this level. We simply omit the action on the model from our query. A traditional approach to answering such causal queries

is to conduct a randomized experiment. However, conducting an experiment may not always be a possibility. In the case of smoking, it would be unethical to force participants to smoke. Practical limitations may also prevent us from obtaining experimental data; the experiment may be too costly to implement or in fact physically impossible to carry out. Furthermore, an experimental study may not be able to reflect the semantics of an ideal manipulation. As an example, a drug may have side-effects that cause unintended perturbations in the physiology of the patient.

The third and topmost level is the counterfactual level which addresses retrospective questions. The term "counterfactual" originates from the term "counterfactual conditional" which was coined by Goodman (1947). Counterfactuals are not only important for scientific thinking, but for traditional western judicial systems as well. When assessing whether the defendant is at fault for causing harm to the plaintiff for example, a guilty verdict can be drawn only if it is found beyond reasonable doubt that the harm would not have been caused had the defendant not acted as they did. In this instance, we compare the real world, where some action was taken, to an alternative hypothetical world, where no action took place. Lewis (1973b) described this concept using the notion of closest worlds and a formal characterization was provided by Galles and Pearl (1998).

Counterfactuals can be vastly more complicated than a single clause comparing two worlds and they may even involve multiple parallel worlds at once. Counterfactuals encompass both of the lower levels of the hierarchy. An interventional question such as "what happens if a smoker stops smoking?" can be reformulated as "what would have happened had a smoker stopped smoking?". This translation does not work if attempted in the opposite direction. No intervention is able to capture the notion of alternative worlds, where an action is both taken and not-taken at the same time. As an example, a question such as "what would have been the effect of a treatment on a subject that was not treated?" cannot be represented as an interventional question. A study cannot be conducted on subjects that have already received treatment to obtain information about a response had they not received the treatment. This has not stopped researchers from attempting to overcome this issue in practical experiments. For example, once a sufficient time has elapsed, the effects of the treatment may be considered non-existent. Alternatively, the originally treated subjects might be replaced with a completely new group of subjects that are as similar to the original subjects as possible in the aspects relevant to the study.

The last step of a causal analysis is the estimation of the identifiable queries of interest based on the available information. Often the estimators are complicated functionals of the observed probability distribution and there is no standard approach that would be applicable in every scenario. Propensity score matching (Rosenbaum and Rubin, 1983) is a standard technique in the framework of potential outcomes (Rubin, 1974) for estimating causal effects. The technique attempts to account for covariates that affect the probability of an intervention, such as a treatment or a policy decision. A similar technique of inverse probability weighting (Robins, 1986) has led to wide application of the so-called doubly-robust methods (e.g. Funk et al., 2011; Cao et al., 2009; Waernbaum, 2012). Doubly robust methods include a regression

model for the outcome of interest as well as a model for the exposure which protects against model misspecification since only one of the models has to be correctly specified for unbiased estimation. In general, if enough data and expert knowledge are available, it may be feasible to fit a parametric model for the observed joint probability distribution and then use the model to simulate potential values for the causal quantity of interest even outside the potential outcome framework.

In this thesis, we focus on the problem of non-parametric identifiability of causal effects and its generalizations to relevant problems in statistics. The purpose of this thesis is to improve the current methodology in three aspects. First, we aim to improve the availability of identification methods by implementing fundamental algorithms and providing them as free software. One of the main contributions of this thesis is the software package `causaleffect` implemented in the statistical computing language R (R Core Team, 2018). Second, we present tools for improving the clarity and practical usability of the pre-existing methods. Even though completeness has been achieved in many identification procedures, the outputs of the corresponding algorithms are often unnecessarily complicated. It turns out that there often exists an equivalent but simpler expression that describes the same quantity of interest. We present a simplification procedure that attempts to symbolically manipulate these output expressions in order to reach a simpler expression. The simplicity of the expression can be evaluated by various criteria. Third, we present a new method called pruning which seeks to eliminate unrelated variables from the causal model before the identification procedure takes place thereby reducing the complexity of the task and the output.

The thesis is organized as follows. In Chapter 2 we review the most important concepts related to graphs and causal models. Chapter 3 addresses the identification of causal effects. Chapter 4 is dedicated to the contribution of this thesis to the field. Chapter 5 concludes with a discussion.

Chapter 2

Causal models

The origins of causal modeling can be traced back to the framework of structural equation models (SEM) (Haavelmo, 1943) which in turn evolved from path analysis (Wright, 1921). SEMs consist of multiple linear regression equations, where the response of one equation may appear as a predictor in another equation. The goal is to give a causal interpretation to the regression coefficients appearing in these equations (Kline, 2005).

A formal model for causality should capture the principles of directionality and modularity discussed in Chapter 1. To motivate the use of graphs, we begin by considering a simple candidate model for representing causality: a model that consists only of the joint probability distribution of the relevant variables. While this model is certainly capable of capturing uncertainty, it falls short in being able to take the main principles into account in every scenario. We can consider a simple system with three variables, X, Y and Z, such that X is the cause of Z and Z is the cause of Y. In this system, we can represent modularity in a probabilistic model with the property that X and Y are conditionally independent when conditioned on Z. This would mean that it is enough to know the direct causes of Y when predicting its value. However, this independence structure also applies if we reverse the roles of X and Y which means that we are unable to represent directionality in this model.

2.1 Graphs

Graphs have evolved to be the standard tool for representing uncertainty. The power of graphs is evidenced by their popularity in recent history and the formalism of graphical models is employed in many scientific disciplines. In the context of path analysis, graphs are an important tool for path tracing which is a method of calculating relationships between variables in the model (Wright, 1934). In structural equation modeling graphs can be used to visually represent the system of equations. In the advent of computational methods, Bayesian networks were introduced where graphs play a central role, even before a causal interpretation was attributed to the directionality of the edges (Pearl, 1985; Lauritzen and Spiegelhalter, 1988).

Graphs have an inherent connection to joint probability distributions that factorize according

to the structure of the graph. When a distribution agrees with a graph in this way, the graph provides an informative representation of the distribution and allows probabilistic reasoning through its properties. It is perhaps this feature that has led to the formalism of graphical models that encompasses various seemingly unrelated special cases.

Next we present the notation and definitions used for graphs throughout this thesis as outlined in (Koller and Friedman, 2009).

A *graph* is an ordered pair $G = \langle \mathbf{V}, \mathbf{E} \rangle$, where $\mathbf{V}$ and $\mathbf{E}$ are sets such that

$$\mathbf{E} \subseteq \big\{ \{X, Y\} \mid X \in \mathbf{V}, Y \in \mathbf{V}, X \neq Y \big\}.$$

The elements of $\mathbf{V}$ are the *vertices* of G and the elements of $\mathbf{E}$ are the *edges* of G. A graph $F = \langle \mathbf{V}', \mathbf{E}' \rangle$ is a *subgraph* of G if $\mathbf{V}' \subseteq \mathbf{V}$ and $\mathbf{E}' \subseteq \mathbf{E}$ such that for each edge $\{X, Y\} \in \mathbf{E}'$ it hold that $X, Y \in \mathbf{V}'$. This is denoted as $F \subseteq G$. A subgraph such that $\mathbf{V}' = \mathbf{V}$ is called an *edge subgraph*. An *induced subgraph*, induced by a set of vertices $\mathbf{W} \subseteq \mathbf{V}$, is denoted by $G[\mathbf{W}]$. This subgraph retains all edges $\{X, Y\} \in \mathbf{E}$ such that $X, Y \in \mathbf{W}$.

A graph G is *directed* if the set $\mathbf{E}$ is considered to consist of ordered pairs (X, Y) instead of sets $\{X, Y\}$. In a directed graph, vertex Y is a *child* of vertex X if G contains an edge from X to Y, which means that $(X, Y) \in \mathbf{E}$. Similarly, X is a *parent* of Y if $(X, Y) \in \mathbf{E}$. The child-parent relationship is often denoted as $X \to Y$, where X is a parent of Y and Y is a child of X. This can also be denoted as $Y \leftarrow X$. For each vertex X of G the sets of *incoming edges* of X and *outgoing edges* of X contain all edges such that $(Y, X) \in \mathbf{E}$ and $(X, Y) \in \mathbf{E}$, respectively. The graph obtained from G by removing all incoming edges of $\mathbf{X}$ and all outgoing edges of $\mathbf{Z}$ is written as $G_{\overline{\mathbf{X}},\underline{\mathbf{Z}}}$.

In (Koller and Friedman, 2009), a path is a sequence of edges connecting adjacent vertices. Here we define paths as graphs where the set of edges forms such a sequence. Let $n \geq 1$ and $\mathbf{V} = \{V_1, \ldots, V_n\}$. If $n > 1$, then the graph $H = \langle \mathbf{V}, \mathbf{E} \rangle$ is a *path* if

$$\mathbf{E} = \big\{ \{V_1, V_2\}, \{V_2, V_3\}, \ldots, \{V_{n-1}, V_n\} \big\}$$

or if

$$\mathbf{E} = \big\{ \{V_1, V_2\}, \{V_2, V_3\}, \ldots, \{V_{n-1}, V_n\}, \{V_n, V_1\} \big\}.$$

In the first case, H is a path from V_1 to V_n. In the second case H is a *cycle*. If $n = 1$, then $H = \langle \{V_1\}, \emptyset \rangle$ is also a path. A path H is a *directed path* from V_1 to V_n if all of its edges are directed and point to the same direction, which means that either

$$\mathbf{E} = \big\{ (V_1, V_2), (V_2, V_3), \ldots, (V_{n-1}, V_n) \big\}$$

or

$$\mathbf{E} = \big\{ (V_1, V_2), (V_2, V_3), \ldots, (V_{n-1}, V_n), (V_n, V_1) \big\}.$$

If a directed graph G does not contain any cycles, it is *acyclic*. A directed acyclic graph is commonly abbreviated as DAG. Even if the graph itself is directed we may still consider (undirected) paths within it.

Additional useful but slightly more complicated relationships between vertices can be defined via paths in directed graphs. Vertex Y is a *descendant* of X in G if there exists a directed path H from X to Y and $H \subseteq G$. Similarly, X is an *ancestor* of Y in G if there exists a directed path H from X to Y and $H \subseteq G$. For a directed graph $G = \langle \mathbf{V}, \mathbf{E} \rangle$ and a set of vertices $\mathbf{W} \subseteq \mathbf{V}$ the sets $\mathrm{Pa}(\mathbf{W})_G, \mathrm{Ch}(\mathbf{W})_G, \mathrm{An}(\mathbf{W})_G$ and $\mathrm{De}(\mathbf{W})_G$ denote a set that contains $\mathbf{W}$ in addition to its parents, children, ancestors and descendants in G, respectively. It is important to note that these sets also contain their argument for convenience and not to be confused with the sets defined through the ancestral relations alone, such as the set of parents. Conversely, a vertex is not considered a parent of itself or to have any other such relation to itself.

Contrary to usual graph theoretic conventions, we call a vertex without any descendants a *root* (typically referred to as sink). The *root set* of G is the set of all roots of G, which is $\{X \in \mathbf{V} \mid \mathrm{De}(X)_G \setminus \{X\} = \emptyset\}$. The reason for this reversal of the names of sinks and roots is to retain consistency with relevant literature (e.g. Shpitser and Pearl, 2006b) and other important definitions.

Two vertices are said to be *connected* if there exists a path between them. Similarly, a graph $G = \langle \mathbf{V}, \mathbf{E} \rangle$ is *connected* if there exists a path between every pair of vertices $V_i, V_j \in \mathbf{V}$. A *connected component* of G is a connected subgraph of G that is not connected to any additional vertices in G. We also define the set $\mathrm{Co}(\mathbf{W})_G$ to denote the set of vertices that are connected to $\mathbf{W}$ in G via paths where the directionality of the edges is ignored, including $\mathbf{W}$.

When a DAG is considered, we can relate an ordering of its vertices to its topological structure. This is useful especially when a causal interpretation is associated with the graph. A *topological ordering* π of a DAG $G = \langle \mathbf{V}, \mathbf{E} \rangle$ is an ordering of its vertices, such that if X is an ancestor of Y in G then $X < Y$ in π. The subset of vertices that are less than V_j in π is denoted by $V_\pi^{(j-1)}$. An algorithm by Kahn (1962) can be used to derive a topological ordering for any DAG. First, we add the vertices without ancestors to the ordering in any order. At the next stage, we add all vertices such that their parents are already contained in the ordering. This is repeated until every vertex has been included. It should be noted that a DAG may have more than one ordering. As a practical consideration, if an ordering is needed for a procedure, it should be fixed beforehand.

2.2 Probabilistic causal model

The most important object of causal inference and the formal description of causality in this thesis is the *probabilistic causal model* (Pearl, 2009).

Definition 2.2.1 (Probabilistic Causal Model)**.** *A* probabilistic causal model *(PCM) is a quadruple*

$$M = \langle \mathbf{U}, \mathbf{V}, \mathbf{F}, P(\mathbf{u}) \rangle,$$

where

1. $\mathbf{U}$ *is a set of unobserved (exogenous) variables that are determined by factors outside the model.*

2. $\mathbf{V}$ *is a set* $\{V_1, V_2, \ldots, V_n\}$ *of observed (endogenous) variables that are determined by variables in* $\mathbf{U} \cup \mathbf{V}$.

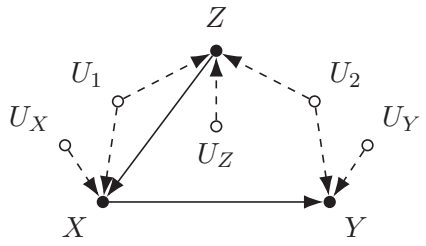
3. $\mathbf{F}$ *is a set of functions* $\{f_{V_1}, f_{V_2}, \ldots, f_{V_n}\}$ *such that each* f_{V_i} *is a mapping from (the respective domains of)* $\mathbf{U} \cup (\mathbf{V} \setminus \{V_i\})$ *to* V_i *and such that the entire set* $\mathbf{F}$ *forms a mapping from* $\mathbf{U}$ *to* $\mathbf{V}$.

4. $P(\mathbf{u})$ *is a joint probability distribution of the variables in the set* $\mathbf{U}$.

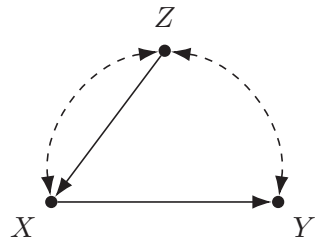
Each causal model induces a causal diagram which is a directed graph that provides a graphical representation of the model. Often we do not have enough information to fully specify the functions of $\mathbf{F}$ and our analysis relies only on the induced graph and the joint probability distribution P. The induced graph contains a vertex for each variable in $\mathbf{U} \cup \mathbf{V}$ and a directed edge from $V_i \in \mathbf{U} \cup \mathbf{V}$ into $V_j \in \mathbf{V}$ whenever f_{V_j} is defined in terms of V_i.

Causal inference often focuses on a sub-class of models that satisfy additional assumptions: each $U \in \mathbf{U}$ appears in at most two functions of $\mathbf{F}$, the variables in $\mathbf{U}$ are mutually independent and the induced graph of the model is acyclic. Models that satisfy these additional assumptions are called *semi-Markovian causal models*. The first two of these assumptions are merely for convenience and do not affect the generality of the presented results. The last assumption of acyclicity is a more straight-forward statement about what type of system we are interested in. Causal models that induce acyclic graphs are called *recursive* (Shpitser and Pearl, 2006b; Kiiveri et al., 1984). The study of non-recursive models that allow feedback loops is also a broad topic of research (e.g. Spirtes, 1995; Koster, 1996; Richardson, 1996; Hyttinen et al., 2012).

A graph associated with a semi-Markovian model is called a *semi-Markovian graph* (SMG). In SMGs every $U \in \mathbf{U}$ has at most two children. When semi-Markovian models are considered we do not depict background variables in the induced graph explicitly. Unobserved variables with exactly two children are not denoted as $V_i \leftarrow U \rightarrow V_j$ but as a bidirected edge $V_i \leftrightarrow V_j$ instead. Furthermore, unobserved variables with only one or no children are omitted entirely. As an example, Figure 2.1(a) shows a graph where every unobserved variable is explicitly visible. Figure 2.1(b) depicts the same graph using the aforementioned abbreviated notation.



(a) A graph showing unobserved variables explicitly.



(b) A graph where unobserved variables are abbreviated.

Figure 2.1: Example on the abbreviations regarding unobserved variables.

These abbreviations provide a useful interpretation where directed edges correspond to causal relations and bidirected edges denote the existence of unobserved confounders. For SMGs the sets $\mathrm{Pa}(\cdot)_G, \mathrm{Ch}(\cdot)_G, \mathrm{An}(\cdot)_G, \mathrm{De}(\cdot)_G$ and $\mathrm{Co}(\cdot)_G$ are defined to contain only observed vertices. Additionally, a subgraph $G[\mathbf{W}]$ of an SMG G will also retain any bidirected edges between vertices in $\mathbf{W}$. Semi-Markovian models are sometimes characterized using *Acyclic directed mixed graphs* (ADMG) (Richardson, 2003). Instead of the notation described for SMGs above, the definition of an ADMG considers directed and bidirected edges explicitly as different entities.

Results concerning semi-Markovian models can be generalized to arbitrary structures of unobserved variables through an operation called *latent projection* (Pearl and Verma, 1991; Verma, 1993).

Definition 2.2.2 (Latent Projection)**.** *Let $G = \langle \mathbf{V} \cup \mathbf{L}, \mathbf{E} \rangle$ be a DAG such that the vertices in $\mathbf{V}$ are observed and the vertices in $\mathbf{L}$ are latent. The* latent projection *$L(G, \mathbf{V})$ is a DAG $\langle \mathbf{V}, \mathbf{E}_L \rangle$, where for every pair of distinct vertices $X, Y \in \mathbf{V}$ it holds that:*

1. *$L(G, \mathbf{V})$ contains an edge $X \to Y$ if there exists a directed path $X \to \cdots \to Y$ in G on which every vertex except X and Y is in $\mathbf{L}$.*

2. *$L(G, \mathbf{V})$ contains an edge $X \leftrightarrow Y$ if there exists a path from X to Y in G that does not contain the pattern $X \to Z \leftarrow Y$ (a collider) and on which every vertex except X and Y is in $\mathbf{L}$ and the first edge has an arrowhead pointing into X and the last edge has an arrowhead pointing into Y.*

From the construction it can be seen that a latent projection of any DAG is in fact an SMG. Furthermore, applying the latent projection to an SMG $G = \langle \mathbf{V}, \mathbf{E} \rangle$ will simply output the graph itself if the vertex set is unchanged, meaning that $L(G, \mathbf{V}) = G$.

Equipped with the causal model we are ready to consider various causal queries. Not every question can be answered by using the assumptions embodied in the induced graph due to the probabilistic nature of the model and presence of confounding. An answer to a causal query in this context can be intuitively understood as a series of operations that enable us to

uniquely compute the query from available information in every model that encodes the same information. Perhaps the most fundamental question in causal inference is whether a given causal query can be uniquely determined in such a manner. A formal characterization of this question is provided by the concept of *identifiability* (Pearl, 2009).

Definition 2.2.3 (Identifiability)**.** *Let* $\mathbf{M}$ *be a set of models with a description* T *and two objects* ϕ *and* θ *computable from each model. Then* ϕ *is* identifiable *from* θ *in* T *if* ϕ *is uniquely computable from* θ *in any model* $M \in \mathbf{M}$*. In other words, all models in* $\mathbf{M}$ *which agree on* θ *also agree on* ϕ*.*

We use the notation of (Shpitser and Pearl, 2008a) and write $T, \theta \vdash_{\text{id}} \phi$ whenever ϕ is identifiable from θ in T. Non-identifiability is denoted as $T, \theta \nvdash_{\text{id}} \phi$. Before we can assess identifiability of various queries, we must draw a connection between the conditional independence properties of the joint distribution P of the causal model and its induced graph G.

2.3 Causal and probabilistic independence

When an induced graph of a causal model is considered, the various paths between variables can be seen to represent the flow of causal influence between them. Direct cause is the simplest form of such flow but information can propagate through directed paths via *mediators* as well. However, our intuition tells us that even if some variables are connected in a network, they do not necessarily influence each other. Sometimes the flow of influence is blocked along such paths. This concept is known as *d-separation* (Pearl, 1986, 1988) of which we present a definition that also accounts for the presence of bidirected edges explicitly (Shpitser and Pearl, 2008a). This definition is called *m-separation* in the context of ADMGs (Richardson, 2003).

Definition 2.3.1 (d-separation)**.** *A path* P *in an SMG* G *is said to be d-separated by a set* $\mathbf{Z}$ *if and only if either*

1. P *contains one of the following three patterns of edges:* $I \to M \to J$*,* $I \leftrightarrow M \to J$ *or* $I \leftarrow M \to J$*, such that* $M \in \mathbf{Z}$ *or*

2. P *contains one of the following three patterns of edges:* $I \to M \leftarrow J$*,* $I \leftrightarrow M \leftarrow J$*,* $I \leftrightarrow M \leftrightarrow J$*, such that* $De(M)_G \cap \mathbf{Z} = \emptyset$*.*

Disjoint sets $\mathbf{X}$ *and* $\mathbf{Y}$ *are said to be d-separated by* $\mathbf{Z}$ *in* G *if every path from* $\mathbf{X}$ *to* $\mathbf{Y}$ *is d-separated by* $\mathbf{Z}$ *in* G*.*

A connection between d-separation in the induced graph G and the conditional independences of the joint probability distribution P can be established because the distribution P of a causal model admits a factorization where each factor corresponds to a conditional distribution of a vertex conditioned on the values of its parents. A probability distribution admitting this factorization is said to be an *independence map* of G (Pearl, 1988).

Definition 2.3.2 (Independence Map)**.** *Let G be the induced graph of a PCM $\langle \mathbf{U}, \mathbf{V}, \mathbf{F}, P(\mathbf{u})\rangle$. Then P is an* independence map *(I-map) of G if*

$$P(\mathbf{v}, \mathbf{u}) = \prod_{\mathbf{v}} P(v_i | Pa^*(v_i)_G \setminus \{v_i\}) \prod_{\mathbf{u}} P(u_i),$$

where $Pa^*(v_i)$ includes the values of the observed and unobserved parents of V_i in G and v_i itself.

An important result of Verma and Pearl (1988) states that if two disjoint sets $\mathbf{X}$ and $\mathbf{Y}$ are d-separated by $\mathbf{Z}$ in G, then $\mathbf{X}$ is independent of $\mathbf{Y}$ given $\mathbf{Z}$ in every distribution P for which G is an I-map. We adopt the standard notation of (Dawid, 1979) and represent conditional independence statements and d-separation of this kind as $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z})_P$ and $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z})_G$, respectively. Remarkably, the conditional independence properties of an SMG G are retained in a latent projection (Verma, 1993). In other words, if we consider a set $\mathbf{W} \subset \mathbf{V}$ latent and it holds that $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z})_G$ for some disjoint subsets $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ of $\mathbf{V} \setminus \mathbf{W}$, then it also holds that $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z})_{L(G, \mathbf{V}\setminus\mathbf{W})}$.

The concept of d-separation can be viewed as causal independence. Furthermore, it allows us to discover probabilistic independences from this causal independence that must hold in every distribution P that is an I-map of G. It is often tempting to make claims in the converse direction as well and infer that there is no causal connection between variables that have been found to be conditionally independent in some distribution. If the distribution at hand is *faithful*, then these kind of propositions are accurate (Pearl, 1988).

Definition 2.3.3 (Faithfulness)**.** *Let G be the induced graph of a PCM $\langle \mathbf{U}, \mathbf{V}, \mathbf{F}, P(\mathbf{u})\rangle$. The distribution P is said to be* faithful *if $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z})_G$ precisely when $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z})_P$ for all disjoint subsets $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ of $\mathbf{V}$.*

In faithful models, d-separation and conditional independence are equivalent which captures the notion that "no correlation implies no causation". A classic pathological example shows how an unfaithful distribution may manifest itself. There are no conditional independences implied by the graph of Figure 2.2.

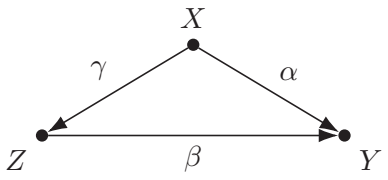


Figure 2.2: An example of a graph where a non-faithful distribution can be constructed.

However, if we consider a linear model where

$$z = \gamma x + \epsilon_z \quad \text{and} \quad y = \alpha x + \beta z + \epsilon_y$$

and ϵ_x, ϵ_y and ϵ_z are independent normally distributed disturbances. In this model the choice of $\alpha = -\beta\gamma$ renders X and Y marginally independent.

Faithfulness is important especially in causal discovery. If the data leads us to conclude some conditional independence relations that are not a result of the true underlying causal structure, it may lead to incorrect conclusions about the learned model. In causal inference, the ramifications are not as severe, but the power of many identifiability results is not as great without the faithfulness assumption. Most identifiability methods are not designed to incorporate extraneous conditional independences that hold in the joint distribution.

The faithfulness assumption is a topic of great debate with both justifications to assume faithfulness in a general non-parametric setting and criticisms warning of the dangers of the assumption (Weinberger, 2018). A measure theoretic argument (e.g. Spirtes et al., 2000) can be used to defend faithfulness on the grounds that cancellation occurs only in a set of parameter values with zero measure. This directly extends to a probability measure over the set of possible parameter values: a set of parameters with zero measure will occur with probability zero.

Steel (2006) further clarifies the measure theoretic argument and shows that only a few fairly general conditions need to hold in order for the argument to be valid. First, the parameters must vary in the joint distribution and they cannot be restricted to discrete sets of values. Second, this must also hold for each parameter when conditioned on the other parameters. This is a formal argument corresponding to the notion that independences of this kind are rare in real world settings since they usually require a very specific parametrization and if we were to slightly perturb the values of the parameters, the cancellation would no longer occur (Pearl, 2009). A counterargument can be made by noting that even if the set of exactly canceling parameters has measure zero, the set of almost canceling parameters need not be and can even have a high probability of occurring (Cartwright, 1999; Hoover, 2001).

Chapter 3

Causal effects

The simplest causal query at the identification level of the causal hierarchy is the causal effect. Causal effects are probability distributions of causal models that are the result of an outside intervention. In an ideal scenario, we are able to make claims regarding this post-interventional model using only the statistical knowledge encoded in the joint probability distribution of the pre-interventional model and the causal knowledge represented by the graph associated with it. In such a setting, causal claims can be made from observational data alone. Recognizing settings where this is possible is often a difficult task and increasingly complex inferential machinery has been developed to address more complicated models.

3.1 Joint and conditional interventional distributions

An important feature of PCMs is their capacity to portray external actions that impose changes to the model. The target of these actions is the set of functions $\mathbf{F}$. The independent causal mechanisms encoded in a PCM correspond to our intuitive notion of modularity. Imposing a change on one function has no effect on any other function. An action that forces a variable $\mathbf{X}$ to take a specific set of values $\mathbf{x}$ is called an *intervention* and it is denoted by $\text{do}(\mathbf{X} = \mathbf{x})$ (Pearl, 2009). The operator $\text{do}(\cdot)$ that performs the intervention is sometimes referred to as the do-operator. An intervention $\text{do}(\mathbf{X} = \mathbf{x})$ on a PCM M creates a new submodel $M_{\mathbf{x}} = \langle \mathbf{V}, \mathbf{U}, \mathbf{F}_{\mathbf{x}}, P(\mathbf{u}) \rangle$, where the new set of functions $\mathbf{F}_{\mathbf{x}}$ is obtained from $\mathbf{F}$ by replacing the functions that determine the value of $\mathbf{X}$ with constant functions that output the corresponding values in $\mathbf{x}$. Other aspects of the submodel, mainly the sets $\mathbf{V}$ and $\mathbf{U}$ and the distribution $P(\mathbf{u})$ are unchanged. The post-interventional model $M_{\mathbf{x}}$ created by the do-operator can be viewed as the result of an ideal manipulation which was discussed in Chapter 1.

This functional change in the model generates a new joint distribution over the observed variables and an induced graph for the submodel $M_{\mathbf{x}}$ that differ from those of the original model M. The *interventional distribution* of a set of variables $\mathbf{Y}$ in the model $M_{\mathbf{x}}$ is denoted by $P_{\mathbf{x}}(\mathbf{Y})$ (or by $P(\mathbf{Y}|\text{do}(\mathbf{X} = \mathbf{x}))$). This distribution is also known as the *causal effect* of $\mathbf{X}$ on $\mathbf{Y}$. For any intervention that we consider, we also require that $P(\mathbf{x}|\text{Pa}(\mathbf{x})_G \setminus \mathbf{x}) > 0$ to ensure that the

full distribution of the submodel $P_{\mathbf{x}}(\mathbf{V})$ is well defined (Pearl, 2009). It should be noted that the term "causal effect" is very conflated in literature and it encompasses many definitions that are not always equivalent. For example, the term is sometimes used to describe the expected difference of a response between treatment and control groups (Rubin, 1974; Holland, 1986). The concept of conditional probability extends naturally to interventional distributions. For disjoint sets $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ conditional interventional distributions or conditional causal effects are of the form $P_{\mathbf{x}}(\mathbf{y}|\mathbf{z})$ and they are defined as

$$P_{\mathbf{x}}(\mathbf{y}|\mathbf{z}) = \frac{P_{\mathbf{x}}(\mathbf{y}, \mathbf{z})}{P_{\mathbf{x}}(\mathbf{z})},$$

whenever $P_{\mathbf{x}}(\mathbf{z}) > 0$. Conditional causal effects can be viewed as a specification of an ordinary intervention. Instead of investigating the effect of an action at the population level, we can target a specific subpopulation of interest. For example, it may be of interest to determine how the level of education affects future salary in a particular age group. By requiring that $\mathbf{Z}$ and $\mathbf{X}$ are disjoint we ensure that the intervention does not create logical contradictions, such as how smoking affects lung cancer among non-smokers.

In addition to joint and conditional interventional distributions it is possible to consider more complicated objects resulting from the effects of actions. Path-specific effects for example relate to settings where we are interested in the effect of $\text{do}(\mathbf{X} = \mathbf{x})$ on $\mathbf{Y}$ along specific paths in the induced graph of the model. In mediation analysis for example, the effect of a treatment is often separated into direct and indirect effects (Robins and Greenland, 1992). We may also consider counterfactual queries by constructing a twin-network graph (Pearl, 2009) or more generally, a counterfactual graph (Shpitser and Pearl, 2007, 2008a).

Typically, we do not have access to the interventional distribution which leaves us with the task of attempting to find a direct link between the joint distribution over the observed variables $P(\mathbf{v})$ and the interventional distribution $P_{\mathbf{x}}(\mathbf{y})$.

3.2 Identifiability of causal effects

We are reliant on the information encoded in the causal model when determining the effects of actions. Causal effects that can be completely characterized by the joint probability distribution of the pre-interventional model and the induced graph fall under the category of identifiable queries as can be seen from the following definition.

Definition 3.2.1 (Causal Effect Identifiability)**.** *Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be an SMG and let $\mathbf{X}$ and $\mathbf{Y}$ be disjoint sets of variables such that $\mathbf{X}, \mathbf{Y} \subset \mathbf{V}$. The causal effect of $\mathbf{X}$ on $\mathbf{Y}$ is said to be* identifiable *from P in G if $P_{\mathbf{x}}(\mathbf{y})$ is uniquely computable from $P(\mathbf{v})$ in any causal model that induces G.*

Recalling Definition 2.2.3, the description T of the set of models $\mathbf{M}$ is now the induced graph G and θ is the joint distribution $P(\mathbf{v})$. The query ϕ to be computed corresponds to the interventional distribution $P_{\mathbf{x}}(\mathbf{y})$. The definition of identifiability gives no direct method

for determining identifiability of a given query. Demonstrating non-identifiability is more straightforward since it is sufficient to construct two PCMs, $M^{(1)}$ and $M^{(2)}$, such that their joint probability distributions agree but the interventional distributions do not. As an example, we show that the causal effect of X on Y is not identifiable in the graph G of Figure 3.1 by constructing two models that generate the same graph but contradicting causal effects. This is the simplest graph where non-identifiability occurs.

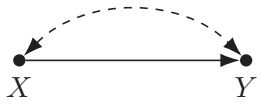


Figure 3.1: An example of a graph where the causal effect of X on Y is not identifiable.

Let U be the unobserved variable corresponding to the bidirected edge between X and Y and let U_Y be an unobserved variable affecting Y. We construct two PCMs, $M^{(1)}$ and $M^{(2)}$, as follows:

$$
\begin{aligned}
&\mathbf{U}^{(1)} = \{U, U_Y\}, && \mathbf{U}^{(2)} = \{U, U_Y\},\\
&\mathbf{V}^{(1)} = \{X, Y\}, && \mathbf{V}^{(2)} = \{X, Y\},\\
&f_X^{(1)}(u) = u, && f_X^{(2)}(u) = u,\\
&f_Y^{(1)}(u, u_y, x) = (1 - u_y)(x \oplus u) + u_y, && f_Y^{(2)}(u, u_y, x) = u_y,\\
&U \sim \mathcal{U}(\{0, 1\}), && U \sim \mathcal{U}(\{0, 1\}),\\
&U_Y \sim \mathcal{U}(\{0, 1\}), && U_Y \sim \mathcal{U}(\{0, 1\}),
\end{aligned}
$$

where $\oplus$ denotes the exclusive or (XOR) operator, $\mathcal{U}(\cdot)$ denotes a uniform distribution and U and U_Y are assumed to be independent in both models. A simple computation shows that the joint distributions of X and Y are:

$P^{(1)}(X,Y)$	Y = 0	Y = 1
X = 0	0.25	0.25
X = 1	0.25	0.25

$P^{(2)}(X,Y)$	Y = 0	Y = 1
X = 0	0.25	0.25
X = 1	0.25	0.25

Under the intervention do$(X = 1)$, the marginal distributions for variable Y update to $P^{(1)}_{x=1}(Y = 1) = 0.75$ and $P^{(2)}_{x=1}(Y = 1) = 0.5$. Similarly, under the intervention do$(X = 0)$ we have that $P^{(1)}_{x=0}(Y = 1) = 0.75$ and $P^{(2)}_{x=0}(Y = 1) = 0.5$. Thus we have that $P^{(1)}(X, Y) = P^{(2)}(X, Y), P^{(1)}(X) = P^{(2)}(X) > 0$ but $P^{(1)}_x(y) \neq P^{(2)}_x(y)$ for both $x = 1$ and $x = 0$ meaning that the effect is not identifiable.

3.3 Manipulation of interventional distributions

An intervention do$(\mathbf{X} = \mathbf{x})$ removes the flow of information from the parents of $\mathbf{X}$ into $\mathbf{X}$ which corresponds to the removal of incoming arrows of $\mathbf{X}$ from the graph in graphical terms. Thus

the induced graph of a submodel $M_{\mathbf{x}}$ is the edge subgraph $G_{\overline{\mathbf{X}}}$ which is sometimes referred to as the *mutilated graph*. This graph has its own set of d-separation statements which in turn imply conditional independences for the interventional distribution known as *dormant independences* (Shpitser and Pearl, 2008b).

In addition to standard probability calculus, a set consisting of three inference rules called *do-calculus* can be used to manipulate interventional distributions (Pearl, 1995). The correctness of these rules can be established through the properties of the mutilated graph. The rules of do-calculus are:

1. Insertion and deletion of observations:

$$P_{\mathbf{x}}(\mathbf{y}|\mathbf{z},\mathbf{w}) = P_{\mathbf{x}}(\mathbf{y}|\mathbf{w}), \text{ if } (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}}}}.$$

2. Exchange of actions and observations:

$$P_{\mathbf{x},\mathbf{z}}(\mathbf{y}|\mathbf{w}) = P_{\mathbf{x}}(\mathbf{y}|\mathbf{z},\mathbf{w}), \text{ if } (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}},\underline{\mathbf{Z}}}}.$$

3. Insertion and deletion of actions:

$$P_{\mathbf{x},\mathbf{z}}(\mathbf{y}|\mathbf{w}) = P_{\mathbf{x}}(\mathbf{y}|\mathbf{w}), \text{ if } (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}},\overline{Z(\mathbf{W})}}},$$

   where $Z(\mathbf{W}) = \mathbf{Z} \setminus \mathrm{An}(\mathbf{W})_{G_{\overline{\mathbf{X}}}}$. In other words, $Z(\mathbf{W})$ is the set of those vertices of $\mathbf{Z}$ that are not ancestors of any vertex in $\mathbf{W}$ in $G_{\overline{\mathbf{X}}}$.

Rule 1 describes simple conditional independence statements in the mutilated graph. When rule 2 is applicable, an intervention $\mathrm{do}(\mathbf{Z} = \mathbf{z})$ has the same effect on $\mathbf{Y}$ as the passive observation $\mathbf{Z} = \mathbf{z}$ has. Rule 3 characterizes situations where adding or removing specific interventions has no effect on $\mathbf{Y}$. We refer to the three rules of do-calculus as rules 1, 2 and 3 throughout this thesis. An alternative approach to do-calculus based on formal logic was proposed by Halpern (2000). In this framework, identifiable causal effects are essentially considered as provable theorems in an axiom system.

Despite its prowess, there are several challenges in applying algebraic methods such as do-calculus in practice. Even though do-calculus has been shown to be complete for identifying causal effects and conditional causal effects (Huang and Valtorta, 2006b; Shpitser and Pearl, 2006b,a), this result is not immediately apparent from the rules themselves. Furthermore, it is not obvious in which order the rules should be applied to find an expression for $P_{\mathbf{x}}(\mathbf{y})$ in terms of $P(\mathbf{v})$.

3.4 Graphical criteria for identifiability

Expressing the interventional distribution $P_{\mathbf{x}}(\mathbf{y})$ in terms of the joint distribution $P(\mathbf{v})$ is a difficult task in general. An effective strategy for characterizing identifiable effects is to

recognize common features between PCMs that produce identifiable effects. These features essentially serve as shortcuts that allow us to bypass the use of do-calculus in the identification task and to obtain the identifying expression directly. Furthermore, models where such a computation is possible can be described via graphical criteria, which typically consist of a set or sets of vertices that block paths responsible for confounding between $\mathbf{X}$ and $\mathbf{Y}$ when adjusted for. The concept of a *back-door path* is important for understanding confounding bias.

Definition 3.4.1 (Back-door path)**.** *Let G be an SMG and let (X, Y) be an ordered pair of vertices of G. A path P is said to be a* back-door path *from X to Y if it contains an edge with an arrowhead pointing into X.*

A simple criterion can be derived from do-calculus directly. If $(\mathbf{Y} \perp\!\!\!\perp \mathbf{X})_{G_{\underline{\mathbf{X}}}}$ then rule 2 applies and we have that $P_{\mathbf{x}}(\mathbf{y}) = P(\mathbf{y}|\mathbf{x})$. This corresponds to a situation where there are no back-door paths between $\mathbf{X}$ and $\mathbf{Y}$. Similarly, if $(\mathbf{Y} \perp\!\!\!\perp \mathbf{X})_{G_{\overline{\mathbf{X}}}}$ then rule 3 applies and $P_{\mathbf{x}}(\mathbf{y}) = P(\mathbf{y})$ which means that there are no directed paths from $\mathbf{X}$ to $\mathbf{Y}$. A simple graphical test described in (Pearl, 1993) can be used to determine whether a suitable set of vertices $\mathbf{Z}$ exists for blocking all back-door paths and rendering $P_{\mathbf{x}}(\mathbf{y})$ identifiable. If a set of vertices $\mathbf{Z}$ blocks all back-door paths from $\mathbf{X}$ to $\mathbf{Y}$ and no member of $\mathbf{Z}$ is a descendant of $\mathbf{X}$ then we obtain

$$P_{\mathbf{x}}(\mathbf{y}) = \sum_{\mathbf{z}} P(\mathbf{y}|\mathbf{z}, \mathbf{x}) P(\mathbf{z}).$$

This result is known as the *back-door criterion* which is often sufficient when all confounders are observed. The set $\mathbf{Z}$ is often not unique and there exist multiple sets that block all back-door paths between $\mathbf{X}$ and $\mathbf{Y}$. For example, in the graph of Figure 3.2 there are four such sets: $\{Z_1, Z_2\}$, $\{Z_2, Z_3\}$, $\{Z_2, Z_4\}$ and $\{Z_2, Z_5\}$. Importantly, the set $\{Z_2\}$ is not sufficient for blocking all back-door paths from X to Y, since conditioning on Z_2 unblocks the path

$$X \leftarrow Z_1 \leftarrow Z_4 \rightarrow Z_2 \leftarrow Z_5 \rightarrow Z_3 \rightarrow Y,$$

where Z_2 is a collider.

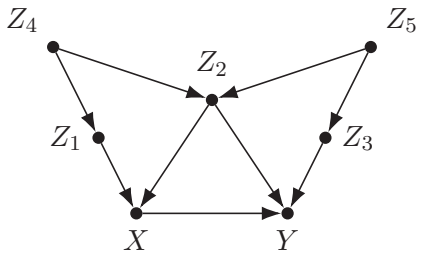


Figure 3.2: An example of a graph where there are multiple sets that block all back-door paths from X to Y.

It is not always possible to find a suitable set $\mathbf{Z}$ for adjustment in the presence of unobserved confounders. Identifiability can still be reached in such a setting sometimes by again finding a

suitable set $\mathbf{Z}$ such that it intercepts all directed paths from $\mathbf{X}$ to $\mathbf{Y}$, there is no back-door path from $\mathbf{X}$ to $\mathbf{Z}$ and all back-door paths from $\mathbf{Z}$ to $\mathbf{Y}$ are blocked by $\mathbf{X}$. If such a set exists we have

$$P_{\mathbf{x}}(\mathbf{y}) = \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}) \sum_{\mathbf{x}'} P(\mathbf{y}|\mathbf{z}, \mathbf{x}')P(\mathbf{z}).$$

This is known as the *front-door criterion* (Pearl, 1995). The idea here is to divide the flow of information from $\mathbf{X}$ to $\mathbf{Y}$ into the flow from $\mathbf{X}$ to $\mathbf{Z}$ and $\mathbf{Z}$ to $\mathbf{Y}$. This enables $\mathbf{X}$ to block the back-door paths from $\mathbf{Z}$ to $\mathbf{Y}$. Figure 3.3 depicts a graph where the front-door criterion applies with $\mathbf{Z} = \{Z_1, Z_2\}$. It is easy to verify that neither $\{Z_1\}$ nor $\{Z_2\}$ alone is able to satisfy the criterion in this graph.

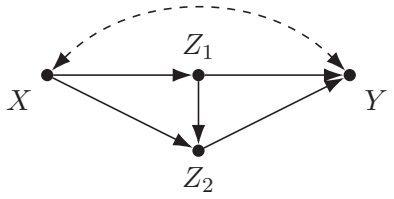


Figure 3.3: An example of a graph where the front-door criterion applies.

More advanced criteria have been derived for SMGs (Galles and Pearl, 1995; Kuroki and Miyakawa, 1999; Tian and Pearl, 2002a), but also for a larger class of models (Perković et al., 2015) and for settings where selection bias is present (Correa and Bareinboim, 2017; Correa et al., 2018). Despite the existence of powerful criteria, they are not sufficient for determining identifiability in all cases. If a specific graphical criterion is not applicable, it does not follow that a causal effect is not identifiable as it may be identifiable by some other means. This is the primary motivation behind the development of identifiability algorithms that completely characterize the problem of computing causal effects of the form $P_{\mathbf{x}}(\mathbf{y})$ and $P_{\mathbf{x}}(\mathbf{y}|\mathbf{z})$.

3.5 Identifiability algorithms

A complete identifiability algorithm always produces a formula for a causal effect of interest in terms of $P(\mathbf{v})$ whenever the effect is identifiable and fails to do so for any non-identifiable effect. Such an algorithm was first proposed by Tian (2002) and studied further by Tian and Pearl (2003) who relied heavily on earlier work on graphical criteria in (Tian and Pearl, 2002a) and the structure known as a c-component studied in (Tian and Pearl, 2002b). This algorithm was independently shown to be complete by Huang and Valtorta (2006b) and Shpitser and Pearl (2006b). Furthermore, it was confirmed by Huang and Valtorta (2006a) that the latent projection of Definition 2.2.2 preserves identifiability, meaning that this algorithm can also be applied to a class of models with arbitrary latent variables by first applying the latent projection.

In this section, we review the most important definitions that led to the conception of the algorithmic approach and present the original algorithm by Tian (2002) as well as the algorithm by Shpitser and Pearl (2006b) which was used to show completeness. An important concept for both algorithms is the set of components of the graph characterized by its bidirected edges.

Definition 3.5.1 (c-component)**.** *Let G be an SMG and let $C \subseteq G$. If every pair of vertices in C is connected by a path consisting entirely of bidirected edges, then C is a* c-component *(confounded component). Furthermore, C is a* maximal c-component *if C contains every vertex connected to C via bidirected paths and C is an induced subgraph of G.*

It should be noted that a graph that has only one vertex is always a c-component, although not necessarily maximal. Any SMG G can always be partitioned into a unique set of maximal c-components, denoted by $C(G)$. This partition can be obtained by first considering an edge subgraph H of G that only contains the bidirected edges of G. The maximal c-components are now the connected components of H. The maximal c-components of an SMG $G = \langle \mathbf{V}, \mathbf{E} \rangle$ provide the so-called c-component factorization of the joint distribution $P(\mathbf{v})$. For any set $\mathbf{C} \subseteq \mathbf{V}$ we define the following function

$$Q[\mathbf{C}](\mathbf{v}) = P_{\mathbf{v}\setminus\mathbf{c}}(\mathbf{c}) = \sum_{\mathbf{u}} \prod_{\mathbf{c}} P(v_i | \mathrm{Pa}^*(v_i) \setminus \{v_i\}) P(\mathbf{u}), \tag{3.1}$$

where $\mathrm{Pa}^*(v_i)$ includes the observed and unobserved parents of V_i and v_i itself. By definition we have that $Q[\mathbf{V}](\mathbf{v}) = P(\mathbf{v})$. For $\mathbf{C} = \emptyset$, we set $Q[\emptyset](\mathbf{v}) = 1$. For the sake of convenience, we abbreviate $Q[\mathbf{C}](\mathbf{v})$ simply as $Q[\mathbf{C}]$. The right-hand side of (3.1) is also known as the *g-formula* (Robins, 1986). The distribution $P(\mathbf{v})$ can now be factorized as

$$P(\mathbf{v}) = \prod_{G[\mathbf{S}_i] \in C(G)} Q[\mathbf{S}_i]. \tag{3.2}$$

The factors $Q[\mathbf{S}_i]$ are called *c-factors* since they correspond to the c-components of G. It was shown by Tian (2002) that $Q[\mathbf{C}]$ is identifiable when $\mathbf{C}$ is a maximal c-component of G. Applying the factorization of (3.2) to the interventional distribution $P_{\mathbf{x}}(\mathbf{y})$ we obtain the *c-component factorization* (Shpitser and Pearl, 2006b)

$$P_{\mathbf{x}}(\mathbf{y}) = \sum_{\mathbf{v}\setminus(\mathbf{y}\cup\mathbf{x})} \prod_{G[\mathbf{S}_i] \in C(G[\mathbf{V}\setminus\mathbf{X}])} Q[\mathbf{S}_i]. \tag{3.3}$$

This factorization is one of the core ideas behind the algorithms of Tian and Pearl (2003) and Shpitser and Pearl (2006b). The original identification task can be recursively broken down into smaller subproblems, mainly the identification of the c-factors appearing on the right-hand side of (3.3), which correspond to the c-components of the subgraph $G[\mathbf{V} \setminus \mathbf{X}]$. Eventually, every subproblem is solved or one of them fails rendering the original problem non-identifiable.

Algorithm 1 is the original identifiability algorithm of Tian (2002) as formulated by Shpitser and Pearl (2006b). The algorithm removes non-ancestors of $\mathbf{Y}$ on line 1 and computes the

C-components of the resulting graph of line 2. This is followed by the c-component factorization on line 3, where Algorithm 2 is used to determine if the corresponding c-factors are identifiable.

Algorithm 1 The causal effect of intervention $do(\mathbf{X} = \mathbf{x})$ on $\mathbf{Y}$ (identify).

INPUT: Value assignments $\mathbf{x}$ and $\mathbf{y}$, joint distribution $P(\mathbf{v})$ and an SMG $G = \langle \mathbf{V}, \mathbf{E} \rangle$. G is an I-map of P.
OUTPUT: Expression for $P_{\mathbf{x}}(\mathbf{y})$ in terms of $P(\mathbf{v})$ or **FAIL**.

function identify$(\mathbf{y}, \mathbf{x}, P, G)$
1: **let** $\mathbf{D} = \mathrm{An}(\mathbf{Y})_{G_{\underline{\mathbf{X}}}}$
2: **assume** $C(G[\mathbf{D}]) = \{G[\mathbf{D}_1], \ldots, G[\mathbf{D}_k]\}, C(G) = \{G[\mathbf{C}_1], \ldots, G[\mathbf{C}_m]\}$
3: **return** $\sum_{\mathbf{D} \setminus \mathbf{Y}} \prod_{i=1}^{k}$ **c-identify**$(G[\mathbf{D}_i], G[\mathbf{C}_j], Q[\mathbf{C}_j])$
where $D_i \subseteq C_j$ **for all** $i = 1, \ldots, k$

Algorithm 2 takes as input two c-components, F and H, where the c-factor $Q[\mathbf{T}]$ corresponding to H is known to be identifiable. There are three possible outcomes regarding the identifiability of $Q[\mathbf{C}]$: either $Q[\mathbf{T}]$ is identifiable from $Q[\mathbf{T}]$ directly on line 1, the identification fails on line 2 or a reduced identification task is solved on line 3. It should be noted that even though Algorithm 2 may seem simple, it is actually quite abstract since it attempts to express the c-factors given as input in terms of other c-factors that are known to be identifiable. Auxiliary results such as Lemma 11 of (Tian, 2002) have to be employed to express an identifiable c-factor in terms of $P(\mathbf{v})$.

Algorithm 2 Computing c-factor $Q[\mathbf{C}]$ from $Q[\mathbf{T}]$ (c-identify).

INPUT: c-components $F = \langle \mathbf{C}, \mathbf{E}_F \rangle, H = \langle \mathbf{T}, \mathbf{E}_H \rangle, F \subseteq H$, probability distribution $Q[\mathbf{T}]$.
OUTPUT: Expression for $Q[\mathbf{C}]$ in terms of $Q[\mathbf{T}]$ or **FAIL**.

function c-identify$(F, H, Q[\mathbf{T}])$
let $\mathbf{A} = \mathrm{An}(\mathbf{C})_H$
1: **if** $\mathbf{A} = \mathbf{C}$, **return** $\sum_{\mathbf{t} \setminus \mathbf{c}} Q[\mathbf{t}]$
2: **if** $\mathbf{A} = \mathbf{T}$, **throw FAIL**
3: **if** $\mathbf{C} \subset \mathbf{A} \subset \mathbf{T}$
let $H[\mathbf{T}'] \in C(H[\mathbf{A}])$ **such that** $\mathbf{C} \subseteq \mathbf{T}' \subseteq \mathbf{A}$
return c-identify$(F, H[\mathbf{T}'], Q[\mathbf{T}'])$
($Q[\mathbf{T}']$ is known to be computable from $\sum_{\mathbf{t} \setminus \mathbf{a}} Q[\mathbf{T}]$.)

An alternative approach to determining identifiability relies on a graphical structure called a *hedge*, which is formed by a pair of special c-components called *c-forests*. To better understand the intuition behind c-trees and hedges, we first present *c-trees*.

Definition 3.5.2 (c-tree)**.** *Let G be a c-component such that every observed vertex has at most*

one child. If there is a vertex Y such that $G[An(Y)_G] = G$, then G is a Y-rooted c-tree.

Identifiability of *direct effects* is linked to c-trees. If a subgraph of G is a Y-rooted c-tree, then the causal effect of $\text{Pa}(Y)$ on Y is not identifiable in G from $P(\mathbf{V})$ (Shpitser and Pearl, 2008a). This shows that identification on a single variable can be difficult and is characterized through a graphical structure. A multivariate generalization of c-trees is the c-forest.

Definition 3.5.3 (c-forest)**.** *Let G be a c-component and let $\mathbf{Y}$ be the root set of G. If every observed vertex of G has at most one child, then G is a $\mathbf{Y}-$*rooted c-forest*.*

It turns out that result regarding direct effects in the case of c-trees cannot be generalized directly to c-forests. What is required is a more complicated structure that consists of a pair of c-forests.

Definition 3.5.4 (hedge)**.** *Let $\mathbf{X}, \mathbf{Y} \subset \mathbf{V}$ be disjoint sets of variables and let G be an SMG. Let $F = \langle \mathbf{V}_F, \mathbf{U}_F, \mathbf{E}_F \rangle$ and $F' = \langle \mathbf{V}_{F'}, \mathbf{U}_{F'}, \mathbf{E}_{F'} \rangle$ be $\mathbf{R}$-rooted c-forests in G such that $\mathbf{V}_F \cap \mathbf{X} \neq \emptyset$, $\mathbf{V}_{F'} \cap \mathbf{X} = \emptyset$, $F' \subseteq F$ and $\mathbf{R} \subseteq An(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$. Then F and F' form a hedge for $P_{\mathbf{x}}(\mathbf{y})$ in G.*

It is not easy to understand hedges intuitively. They can be used to construct a pair of causal models for a given identifiability problem, where the interventional distributions do not agree despite the fact that the models have the same joint probability distribution over the observed variables. Shpitser and Pearl (2006b) showed that hedges completely characterize the identifiability of causal effects and provided Algorithm 3, known as the ID algorithm, that simultaneously tries to derive an expression for the effect and keeps track of the structure of the graph in an attempt to detect the presence of a possible hedge. They also showed soundness and completeness of the algorithm: a correct expression is always returned for an identifiable causal effect, otherwise a hedge witnessing non-identifiability is returned. Incidentally, this also shows the completeness of do-calculus, since each line of Algorithm 3 corresponds to standard probability manipulations and application of the rules of do-calculus.

Algorithm 3 The causal effect of intervention $do(\mathbf{X} = \mathbf{x})$ on $\mathbf{Y}$ (ID).

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

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

function ID$(\mathbf{y}, \mathbf{x}, P, G)$

1: **if** $\mathbf{x} = \emptyset$,
  return $\sum_{\mathbf{v} \setminus \mathbf{y}} P(\mathbf{v})$.

2: **if** $\mathbf{V} \neq \mathrm{An}(\mathbf{Y})_G$,
  return ID$(\mathbf{y}, \mathbf{x} \cap \mathrm{An}(\mathbf{Y})_G, P(\mathrm{An}(\mathbf{Y})_G), G[\mathrm{An}(\mathbf{Y})_G])$.

3: **let** $\mathbf{W} = (\mathbf{V} \setminus \mathbf{X}) \setminus \mathrm{An}(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$.
  if $\mathbf{W} \neq \emptyset$,
  return ID$(\mathbf{y}, \mathbf{x} \cup \mathbf{w}, P, G)$.

4: **if** $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}_1], \ldots, G[\mathbf{S}_k]\}$,
  return $\sum_{\mathbf{v} \setminus (\mathbf{y} \cup \mathbf{x})} \prod_{i=1}^{k}$ **ID**$(\mathbf{s}_i, \mathbf{v} \setminus \mathbf{s}_i, P, G)$.
  if $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}]\}$,

5: **if** $C(G) = \{G\}$,
  throw FAIL$(G, G[\mathbf{S}])$.

6: **if** $G[\mathbf{S}] \in C(G)$,
  return $\sum_{\mathbf{s} \setminus \mathbf{y}} \prod_{V_i \in \mathbf{S}} P(v_i | v_\pi^{(i-1)})$.

7: **if** $(\exists \mathbf{S}')\mathbf{S} \subset \mathbf{S}'$ **such that** $G[\mathbf{S}'] \in C(G)$,
  return ID$(\mathbf{y}, \mathbf{x} \cap \mathbf{s}', \prod_{V_i \in \mathbf{S}'} P(V_i | V_\pi^{(i-1)} \cap \mathbf{S}', v_\pi^{(i-1)} \setminus \mathbf{s}'), G[\mathbf{S}'])$.

Using their previous work as a foundation, Shpitser and Pearl (2006a) further showed that the problem of identifying conditional causal effects can also be solved using hedges. They provided Algorithm 4, known as IDC, which can be used to determine the identifiability of a given conditional causal effect.

Algorithm 4 The causal effect of intervention $do(\mathbf{X} = \mathbf{x})$ on $\mathbf{Y}$ given $\mathbf{Z}$ (IDC).

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

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

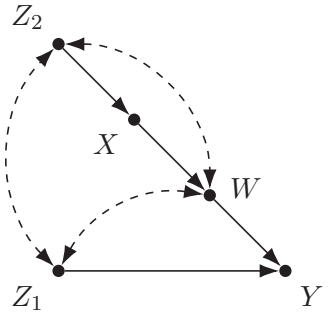
function IDC$(\mathbf{y}, \mathbf{x}, \mathbf{z}, P, G)$

1: **if** $\exists Z \in \mathbf{Z}$ such that $(\mathbf{Y} \perp\!\!\!\perp Z | \mathbf{X}, \mathbf{Z} \setminus \{Z\})_{G_{\overline{\mathbf{X}}, \underline{\mathbf{Z}}}}$,
  return IDC$(\mathbf{y}, \mathbf{x} \cup \{z\}, \mathbf{z} \setminus \{z\}, P, G)$.

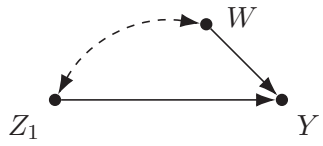
2: **else let** $P' =$ **ID**$(\mathbf{y} \cup \mathbf{z}, \mathbf{x}, P, G)$.
  return $P' / \sum_{\mathbf{y}} P'$.

To illustrate how Algorithms 1 and 3 work, we compare their operation when the causal

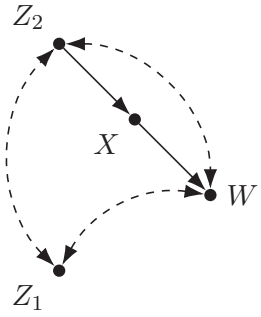
effect of X on Y is computed in the graph G of Figure 3.4(a). We fix the topological ordering π of G as $Y > W > X > Z_2 > Z_1$.



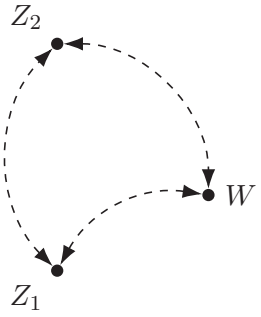
(a) Graph G.



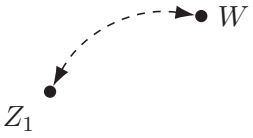
(b) Subgraph $G[\mathbf{D}]$.



(c) Subgraph $G^{(1)}$.



(d) Subgraph $G^{(2)}$.



(e) Subgraph $G^{(3)}$.

Figure 3.4: Subgraphs used in the computation of $P_x(y)$ for the example on the operation of Algorithms 1 and 3.

In the case of the ID algorithm, the first call ID$(\{y\}, \{x\}, P, G)$ brings us to line 3, where

$$\mathbf{W} = (\mathbf{V} \setminus \mathbf{X}) \setminus \mathrm{An}(\mathbf{Y})_{G_{\overline{X}}} = (\{Y, W, X, Z_2, Z_1\} \setminus \{X\}) \setminus \{Y, W, X, Z_1\} = \{Z_2\}.$$

We add Z_2 to the intervention set and call ID$(\{y\}, \{x, z_2\}, P, G)$. The computation continues on line 4, since

$$C(G[\mathbf{V} \setminus \mathbf{X}]) = C(G[Y, W, Z_1]) = \{G[Y], G[W, Z_1]\},$$

we have that

$$P_x(y) = P_{x,z_2}(y) = \sum_{w,z_1} P_{w,x,z_2,z_1}(y) P_{y,x,z_2}(z_1, w), \tag{3.4}$$

which requires the evaluation of two recursive calls corresponding to the two c-components: ID$(\{y\}, \{w, x, z_2, z_1\}, P, G)$ and ID$(\{z_1, w\}, \{y, x, z_2\}, P, G)$. The first recursive call ends up

on line 6 since the c-component $G[Y]$ of $G[Y, W, Z_1]$ is also a c-component of G. It follows that

$$P_{w,x,z_2,z_1}(y) = P(y|z_1, z_2, x, w). \tag{3.5}$$

The second recursive call ends up on line 2 because Y is not an ancestor of Z_1 or W. By calling $\text{ID}(\{z_1, w\}, \{x, z_2\}, P^{(1)}, G^{(1)})$ the computation continues in the subgraph $G^{(1)} = G[W, X, Z_2, Z_1]$ shown in Figure 3.4(c) with the distribution $P^{(1)} = P(W, X, Z_2, Z_1)$. This time line 7 is triggered by noting that

$$C(G^{(1)}[\mathbf{V} \setminus \mathbf{X}]) = C(G^{(1)}[W, Z_1]) = \{G^{(1)}[W, Z_1]\},$$

and that $\mathbf{S} = \{W, Z_1\} \subset \{W, Z_2, Z_1\} = \mathbf{S}'$ where $G^{(2)} = G^{(1)}[\mathbf{S}']$ is a c-component of $G^{(1)}$ as shown in Figure 3.4(d) . The next recursive call is $\text{ID}(\{z_1, w\}, \{z_2\}, P^{(2)}, G^{(2)}[\mathbf{S}'])$ where

$$P^{(2)} = \prod_{V_i \in \mathbf{S}'} P^{(1)}(V_i|V_\pi^{(i-1)} \cap \mathbf{S}', v_\pi^{(i-1)} \setminus \mathbf{s}') = P(W|Z_1, Z_2, x)P(Z_2|Z_1)P(Z_1).$$

Line 2 is reached again due to Z_2 not being an ancestor of Z_1 or W in $G^{(2)}$ resulting in the call $\text{ID}(\{z_1, w\}, \emptyset, P^{(3)}, G^{(3)})$ with $G^{(3)} = G^{(2)}[Z_1, W]$, shown in Figure 3.4(e) and $P^{(3)} = \sum_{z_2} P^{(2)}$. Line 1 is finally reached since the intervention set is empty and we obtain

$$P_{y,x,z_2}(z_1, w) = \sum_{z_2} P(w|z_1, z_2, x)P(z_2|z_1)P(z_1). \tag{3.6}$$

Inserting the expressions (3.5) and (3.6) for the causal effects $P_{w,x,z_2,z_1}(y)$ and $P_{y,x,z_2}(z_1, w)$ back into the expression (3.4) for $P_x(y)$ yields an expression for the causal effect:

$$P_x(y) = \sum_{w,z_1} P(y|z_1, z_2, x, w) \sum_{z_2'} P(w|z_1, z_2', x)P(z_2'|z_1)P(z_1).$$

For comparison, the Algorithm 1 starts by computing the subgraph $G[\mathbf{D}] = G[\text{An}(Y)_{G_{\underline{X}}}]$ depicted in Figure 3.4(b). Next, the set of c-components of $G[\mathbf{D}]$ is derived on line 2 and it is

$$C(G[\mathbf{D}]) = C(G[Y, W, Z_1]) = \{G[Y], G[W, Z_1]\}.$$

The set of C-components of G is also derived and it is

$$C(G) = \{G[Y], G[X], G[W, Z_2, Z_1]\}.$$

Next, recursive calls to C-IDENTIFY are launched to compute the following expression on line 3

$$P_x(y) = \sum_{w,z_1} Q[Y]Q[W, Z_1] = \sum_{w,z_1} P_{w,x,z_2,z_1}(y)P_{y,x,z_2}(z_1, w),$$

which is the same expression as derived by the ID algorithm for this causal effect. We proceed to identify the c-factor $Q[Y]$ by calling C-IDENTIFY$(G[Y], G[Y], Q[Y])$. We compute the set

$\mathbf{A} = \text{An}(Y)_{G[Y]} = \{Y\}$ and since $\mathbf{A} = \{Y\} = \mathbf{C}$ we know this c-factor to be identifiable. Fortunately, we already know that the expression is (3.5) from the computation using the ID algorithm and it is

$$Q[Y] = P(y|z_1, z_2, x, w).$$

Alternatively, lemma 11 of (Tian, 2002) could have been used for manual derivation of this c-factor. Next, we attempt to identify the c-factor $Q[W, Z_1]$ from $Q[W, Z_2, Z_1]$ by initiating a call to C-IDENTIFY$(G[W, Z_1], G[W, Z_2, Z_1], Q[W, Z_2, Z_1])$, since $\{W, Z_1\} \subset \{W, Z_2, Z_1\}$. Again, we compute the set $\mathbf{A} = \text{An}(W, Z_1)_{G[W,Z_2,Z_1]} = \{W, Z_1\}$. We have that $\mathbf{A} = \{W, Z_1\} = \mathbf{C}$ and obtain

$$Q[W, Z_1] = \sum_{z_2} Q[W, Z_2, Z_1].$$

Once more, we know this c-factor to be identifiable and have already obtained its expression 3.6 in the computation using the ID algorithm

$$Q[W, Z_1] = \sum_{z_2} P(w|z_1, z_2, x)P(z_2|z_1)P(z_1).$$

Here we used the results obtained from the derivation through the ID algorithm to compute the expressions of the required c-factors. This illustrates the difficulty of applying Algorithm 1 in practice. Without pre-existing knowledge, a manual derivation or another algorithm is required to derive a suitable expression for the identifiable c-factors. This issue is entirely bypassed by the ID algorithm.

Chapter 4

Research contribution

This section summarizes the research contribution of Articles I-III to the causal inference methodology.

4.1 Implementation of causal inference algorithms

Article I describes an implementation of Algorithms 3 and 4 in a software package called `causaleffect` using the statistical computing language R (R Core Team, 2018). The article illustrates how these causal effects and conditional causal effects can be identified in practice and how to construct a graph of interest. An in-depth description of the implementation itself is also given which may be of use for subsequent implementations or other works taking advantage of the `causaleffect` package. A substantial review of R packages related to causality was also done to discover the extent of how R can be used for causal inference. The R language was chosen for the implementation for various reasons. First, R is freely available for almost any platform and it is still being actively developed. Second, the R language has a wide reach through the Comprehensive R Archive Network (CRAN) which hosts the majority of R packages and connects package developers to the users effortlessly. R users can always obtain the most up-to-date version of their favorite packages through CRAN. Finally, the implementation does not have to be built from scratch, since a powerful library for constructing and manipulating graphs called `igraph` (Csardi and Nepusz, 2006) could be taken advantage of.

Since the publication of Article I, the `causaleffect` package has received new features in addition to those described in Articles II and III. The package currently also implements algorithms for z-identifiability (Bareinboim and Pearl, 2012a), meta-transportability (Bareinboim and Pearl, 2014) and selection bias recoverability (Bareinboim and Tian, 2015). The package can also be used to discover functional constraints of causal models using a systematic method of (Tian and Pearl, 2002b).

4.2 Simplification of causal effect formulas

Article II provides a rigorous approach for expression simplification in a probabilistic context. The fundamental primitive unit of a probabilistic expression is an atomic expression of the form

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

In other words, atomic expressions are marginalized products of conditional distributions where the left-hand side contains a single variable. More complicated expressions can be constructed using atomic expressions.

In the context of the article, the focus is on simplifying expressions obtained as output from Algorithms 3 and 4. It is challenging to give a general definition of simplification. We can, however, consider various criteria when comparing two expressions for the same causal effects of interest, such as expression length, the number of summations, the number of unique variables and the number of fractions. The article provides a complete simplification procedure for atomic expressions that admit a specific factorization in terms of a single variable that is to be eliminated from the expression. This procedure can also be applied to non-atomic expressions by first simplifying all atomic expressions that were used construct the non-atomic expressions, combining the results into new atomic expressions and simplifying them again until the expression can no longer be further simplified. The methods of Article II have been made available through the `causaleffect` package.

4.3 Detecting variables that are unnecessary for identifiability

Article III is in some sense a continuation of Article II. Often complicated expressions arise as a result of variables being included in the causal model that play no role in the identification of causal effect of interest. Due to the nature of the ID algorithm, the presence of these variables carries over to the output which can have detrimental effects. Typically, a simpler expression can easily be derived by removing these variables from the causal model before the identification task takes place. We call this operation pruning.

Article III provides several criteria that can be used to easily construct sets of variables that can be pruned from the model. Even though these methods are not shown to be complete, they have a polynomial time complexity and avoid enumeration over the possible vertex subsets making them suitable for almost any setting. Some examples of prunable sets include specific ancestors of the variable that has been intervened upon and variables that are connected to other vertices through a single vertex. It is also shown that latent projections can be used as a tool for pruning. The pruning methods presented in Article III are combined into a new pruning identifiability algorithm which is again provided through the `causaleffect` package.

Chapter 5

Discussion

The aim of this thesis was to improve the current causal inference methodology by focusing on the practical aspects of algorithmic solutions to the causal effect identifiability problem. We provided the standard methods as a software package that is freely available (Article I). We studied to what extent the output of the identifiability algorithm can be simplified after the identification task has already been completed and the desired expression is obtained (Article II). We also investigated the possible operations to be carried out before the identification task such that a simpler expression could be obtained directly (Article III).

There have been remarkably few software implementations of any causal identification methods. As far as we know, implementations of Algorithms 3 and 4, in addition to the package described in Article I, can only be found in the CIBN software by Jin Tian available at `http://web.cs.iastate.edu/~jtian/Software/CIBN.htm`. More advanced methods listed in Section 4.1 are not publicly available elsewhere to the best of our knowledge. Popular software packages such as `DAGitty` (Textor et al., 2016) and `pcalg` (Kalisch et al., 2012) provide graphical adjustment criteria but lack complete methods which is understandable, since the identifiability of causal effects is not the primary focus of either package. The R package `medflex` (Steen et al., 2017) provides a comprehensive selection of tools for mediation analysis.

The approach to expression simplification proposed in Article II creates pathways for possible future work. Importantly, the proposed simplification algorithm, although complete to a specific class of expressions, has exponential time complexity in the number of vertices of the associated graph. It may be possible to derive an alternative but equivalent procedure that would operate in polynomial time. Further improvements to the efficiency of the procedure could be discovered by considering graphs or expressions with a specific structure. Benefits of such considerations could be realized even if the problem itself turns out to be NP-hard. A reviewer of Article II also suggested that simplification may have a connection with dormant independence, a concept that was briefly mentioned in Section 3.3, which opens up another line of inquiry.

Various sufficient criteria for pruning were discovered in Article III. A complete characterization of the problem is challenging, since some pruning operations are mutually exclusive.

This does not rule out the possibility of deriving stronger sufficient criteria that are also easily applicable in practice without resorting to heavy computation.

Articles II and III share a common theme of equivalent expressions that describe the same causal effect. When two expressions are compared, it is often beneficial to use the simpler of the two especially if it is known that variables affected by bias do not appear in the simpler expression. This notion can be generalized to ask the following question: If a researcher is equipped with a set of expressions for a causal effect of interest, which one should be used? In other words, which expression has the most desirable properties for estimation?

Bibliography

Andreassen, S., Woldbye, M., Falck, B., and Andersen, S. K. (1987). Munin—a causal probabilistic network for interpretation of electromyographic findings. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 366–372. Morgan Kaufmann.

Angrist, J. D., Imbens, G. W., and Rubin, D. B. (1996). Identification of causal effects using instrumental variables. *Journal of the American Statistical Association*, 91(434):444–455.

Bareinboim, E. and Pearl, J. (2012a). Causal inference by surrogate experiments: z-identifiability. In de Freitas, N. and Murphy, K., editors, *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, pages 113–120. AUAI Press.

Bareinboim, E. and Pearl, J. (2012b). Controlling selection bias in causal inference. In Lawrence, N. D. and Girolami, M., editors, *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, volume 22, pages 100–108.

Bareinboim, E. and Pearl, J. (2014). Transportability from multiple environments with limited experiments: Completeness results. In *Proceedings of the 27th International Conference on Neural Information Processing Systems – Volume 1*, pages 280–288. MIT Press.

Bareinboim, E. and Tian, J. (2015). Recovering causal effects from selection bias. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 3475–3481. AAAI Press.

Bareinboim, E., Tian, J., and Pearl, J. (2014). Recovering from selection bias in causal and statistical inference. In *Proceedings of the 28th AAAI Conference on Neural Information Processing Systems*, pages 2410–2416. AAAI Press.

Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of The Royal Society of London*, 53:370–418.

Blalock, H. M. (1971). *Causal Models in the Social Sciences*. Aldine Publishing Company.

Cao, W., Tsiatis, A. A., and Davidian, M. (2009). Improving efficiency and robustness of the doubly robust estimator for a population mean with incomplete data. *Biometrika*, 96(3):723–734.

Cartwright, N. (1999). *The Dappled World: A Study of the Boundaries of Science*. Cambridge University Press.

Chen, B., Kumor, D., and Bareinboim, E. (2017). Identification and model testing in linear structural equation models using auxiliary variables. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 757–766.

Cooper, G. F. (2000). A Bayesian method for causal modeling and discovery under selection. In Boutlier, C. and Goldszmidt, M., editors, *Proceedings of the 16th Conference of Uncertainty in Artificial Intelligence*, pages 98–106. Morgan Kaufmann.

Correa, J. and Bareinboim, E. (2017). Causal effect identification by adjustment under confounding and selection biases. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 3740–3746.

Correa, J., Tian, J., and Bareinboim, E. (2018). Generalized adjustment under confounding and selection biases. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695(5):1–9.

Dawid, A. P. (1979). Conditional independence in statistical theory. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(1):1–31.

Didelez, V., Kreiner, S., and Keiding, N. (2010). Graphical models for inference under outcome-dependent sampling. *Statistical Science*, 25(3):368–387.

Funk, M. J., Westreich, D., Wiesen, C., Stürmer, T., Brookhart, M. A., and Davidian, M. (2011). Doubly robust estimation of causal effects. *American Journal of Epidemiology*, 173(7):761–767.

Galles, D. and Pearl, J. (1995). Testing identifiability of causal effects. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 185–195. Morgan Kaufmann.

Galles, D. and Pearl, J. (1998). An axiomatic characterization of causal counterfactuals. *Foundations of Science*, 3(1):151–182.

Geneletti, S., Richardson, S., and Best, N. (2009). Adjusting for selection bias in retrospective, case-control studies. *Biostatistics*, 10(1):17–31.

Goodman, N. (1947). The problem of counterfactual conditionals. *The Journal of Philosophy*, 44(5):113–128.

Granger, C. W. J. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37:424–438.

Griffiths, T. L. and Tenenbaum, J. B. (2009). Theory-based causal induction. *Psychological Review*, 116(4):661–716.

Haavelmo, T. (1943). The statistical implications of a system of simultaneous equations. *Econometrica*, 11(1):1–12.

Halpern, J. Y. (2000). Axiomatizing causal reasoning. *Journal of Artificial Intelligence Research*, 12:317–337.

Holland, P. W. (1986). Statistics and causal inference. *Journal of the American Statistical Association*, 81(396):945–960.

Hoover, K. D. (2001). *Causality in Macroeconomics*. Cambridge University Press.

Huang, Y. and Valtorta, M. (2006a). Identifiability in causal Bayesian networks: A sound and complete algorithm. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1149–1154. AAAI Press.

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

Hyttinen, A., Eberhardt, F., and Hoyer, P. O. (2012). Learning linear cyclic causal models with latent variables. *Journal of Machine Learning Research*, 13(1):3387–3439.

Kahn, A. B. (1962). Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562.

Kalisch, M., Mächler, M., Colombo, D., Maathuis, M. H., and Bühlmann, P. (2012). Causal inference using graphical models wit the R package pcalg. *Journal of Statistical Software*, 47(11):1–26.

Karvanen, J. (2015). Study design in causal models. *Scandinavian Journal of Statistics*, 42(2):361–377.

Kiiveri, H., Speed, T. P., and Carlin, J. B. (1984). Recursive causal models. *Journal of the Australian Mathematical Society. Series A. Pure Mathematics and Statistics*, 36(1):30—-52.

Kline, R. B. (2005). *Principles and Practice of Structural Equation Modeling*. Guilford Publications.

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

Koster, J. T. A. (1996). Markov properties of nonrecursive causal models. *The Annals of Statistics*, 24(5):2148–2177.

Kuroki, M. and Miyakawa, M. (1999). Identifiability criteria for causal effects of joint interventions. *Journal of the Japan Statistical Society*, 29(2):105–117.

Laplace, P. S. (1812). *Essai philosophique sur les probabilités*. Courcier Imprimeur.

Larrañaga, P. and Moral, S. (2011). Probabilistic graphical models in artificial intelligence. *Applied Soft Computing*, 11(2):1511–1528.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988). 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.

Lewis, D. (1973a). Causation. *Journal of Philosophy*, 70:556–567.

Lewis, D. (1973b). *Counterfactuals*. Harvard University Press.

Neyman, J. (1923). Sur les applications de la théorie des probabilités aux experiences agricoles: Essai des principes. *Roczniki Nauk Rolniczych*, 10:1–51.

Pearl, J. (1985). A constraint-propagation approach to probabilistic reasoning. In *Proceedings of the 1st Conference on Uncertainty in Artificial Intelligence*. AUAI Press.

Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29:241–288.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Pearl, J. (1993). Comment: Graphical models, causality and intervention. *Statistical Science*, 8:266–269.

Pearl, J. (1995). Causal diagrams for empirical research. *Biometrika*, 82(4):669–688.

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

Pearl, J. and Verma, T. S. (1991). A theory of inferred causation. In *Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference*, pages 441–452.

Perković, E., Textor, J., Kalisch, M., and Maathuis, M. (2015). A complete generalized adjustment criterion. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, pages 682–691. AUAI Press.

Peters, J., Janzing, D., Gretton, A., and Schölkopf, B. (2009). Detecting the direction of causal time series. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 801–808.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing.

Reichenbach, H. (1956). *Direction of Time*. University of California Press.

Richardson, T. S. (1996). *Feedback Models: Interpretation and Discovery*. PhD thesis, Carnegie Mellon University.

Richardson, T. S. (2003). Markov properties for acyclic directed mixed graphs. *Scandinavian Journal of Statistics*, 30(1):145–157.

Robins, J. M. (1986). A new approach to causal inference in mortality studies with a sustained exposure period—application to control of the healthy worker survivor effect. *Mathematical Modelling*, 7:1393–1512.

Robins, J. M. and Greenland, S. (1992). Identifiability and exchangeability for direct and indirect effects. *Epidemiology*, 3(2):143–155.

Robins, J. M. and Wasserman, L. (1999). On the impossibility of inferring causation from association without background knowledge. In Glymour, C. and Cooper, G. F., editors, *Computation, Causation & Discovery*, pages 302–321. AAAI / MIT Press.

Rosenbaum, P. R. and Rubin, D. B. (1983). The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55.

Rubin, D. B. (1974). Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66(5):688–701.

Shpitser, I., Mohan, K., and Pearl, J. (2015). Missing data as a causal and probabilistic problem. In Meila, M. and Heskes, T., editors, *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, pages 802–811. AUAI Press.

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

Shpitser, I. and Pearl, J. (2006b). 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.

Shpitser, I. and Pearl, J. (2007). What counterfactuals can be tested. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 352–359. AUAI Press.

Shpitser, I. and Pearl, J. (2008a). Complete identification methods for the causal hierarchy. *Journal of Machine Learning Research*, 9:1941–1979.

Shpitser, I. and Pearl, J. (2008b). Dormant independence. In *Proceedings of the 23rd National Conference on Artificial Intelligence – Volume 2*, pages 1081–1087. AAAI Press.

Spirtes, P. (1995). Directed cyclic graphical representation of feedback models. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 491–498. Morgan Kaufmann.

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

Steel, D. (2006). Homogeneity, selection, and the faithfulness condition. *Minds and Machines*, 16:303–317.

Steen, J., Loeys, T., Moerkerke, B., and Vansteelandt, S. (2017). medflex: An R package for flexible mediation analysis using natural effect models. *Journal of Statistical Software*, 76(11).

Strotz, R. H. and Wold, H. O. A. (1960). Recursive versus nonrecursive systems: An attempt at synthesis. *Econometrica*, 28(2):417–427.

Textor, J., van der Zander, B., Gilthorpe, M. K., and Liskiewicz, M. (2016). Robust causal inference using directed acyclic graphs: the R package 'dagitty'. *International Journal of Epidemiology*, 45(6):1887–1894.

Thoemmes, F. and Mohan, K. (2015). Graphical representation of missing data problems. *Structural Equation Modeling: A Multidisciplinary Journal*, 22(2):631–642.

Tian, J. (2002). *Studies in Causal Reasoning and Learning*. PhD thesis, Department of Computer Science, University of California, Los Angeles.

Tian, J. and Pearl, J. (2002a). A general identification condition for causal effects. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 567–573. AAAI Press.

Tian, J. and Pearl, J. (2002b). On the testable implications of causal models with hidden variables. In *Proceedings of the 18th Conference of Uncertainty in Artificial Intelligence*, pages 519–527. AUAI Press.

Tian, J. and Pearl, J. (2003). On the identification of causal effects. Technical report, Department of Computer Science, University of California, Los Angeles. R-290-L.

van der Zander, B. and Liskiewicz, M. (2016). On searching for generalized instrumental variables. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*.

Verma, T. and Pearl, J. (1988). Influence diagrams and d-separation. Technical report, Cognitive Systems Laboratory, University of California, Los Angeles. R-101.

Verma, T. S. (1993). Graphical aspects of causal models. Technical report, Department of Computer Science, University of California, Los Angeles. R-191.

Waernbaum, I. (2012). Model misspecification and robustness in causal inference: Comparing matching with doubly robust estimation. *Statistics in Medicine*, 31(15):1572–1581.

Weinberger, N. (2018). Faithfulness, coordination and causal coincidences. *Erkenntnis*, 83(2):113–133.

Wright, S. (1921). Correlation and causation. *Journal of Agricultural Research*, 20(7):557–585.

Wright, S. (1934). The method of path coefficients. *The Annals of Mathematical Statistics*, 5(3):161–215.

Appendix A

Errata for original publications

Article II

1. Pages 17 through 18: The sets being computed should be denoted with bold letters, i.e. $\mathbf{A}$ and $\mathbf{B}$ instead of A and B.

2. Page 17: Computing the set $\mathbf{A} = (\mathrm{An}^*(Z_3) \cup \mathbf{P}_4) \triangle \mathbf{D}$ should result in $\emptyset$ instead of $\{X\}$.

3. Page 27: The numerator $P(\mathbf{V}, \mathbf{M} \setminus \mathbf{M}^-)$ should be $P(\mathbf{V}, \mathbf{M} \setminus \mathbf{M}^- | \mathbf{D})$ instead.

Article III

1. Pages 12 through 13: The term $P(w|\mathrm{Pa}(w)_G \setminus (\mathbf{z} \cup \{w\}))$ should be $P(w|\mathrm{Pa}(w)_G \setminus (\mathbf{z} \cup \mathbf{u}_\mathbf{z} \cup \{w\}))$ instead.

2. Page 13: The proof of Corollary 13 is poorly worded. A reformulated proof shows explicitly why the constructed set $\mathbf{T}$ satisfies the conditions for the set $\mathbf{Z}$ of Theorem 12: Let $\mathbf{R} = \mathrm{An}(W)_{G_{\overline{\mathbf{x}}}} \setminus \mathrm{De}(\mathbf{X})_G$ and let $\mathbf{T} = \mathbf{R} \setminus \mathrm{Co}(\mathbf{V} \setminus \mathbf{R})_{G_{\overline{W}}}$. Since $G = G[\mathrm{An}(\mathbf{Y})_G]$ and the set $\mathbf{R}$ does not contain $\mathbf{X}$ or any of its descendants, we have that $\mathbf{T} \cap (\mathbf{Y} \cup \mathbf{X}) = \emptyset$. The set $\mathrm{Co}(\mathbf{V} \setminus \mathbf{R})_{G_{\overline{W}}}$ contains all vertices that are connected to $\mathbf{V} \setminus \mathbf{R}$ through a path that does not contain W. It follows that the set $\mathbf{T}$ contains only those members of $\mathbf{R}$ that are connected to $\mathbf{V} \setminus \mathbf{R}$ only through W. Thus the set $\mathbf{T}$ satisfies both criteria for the set $\mathbf{Z}$ of Theorem 12.

3. Page 15: $P_x(y)$ should be $P_x(\mathbf{y})$ in the proof of Corollary 16.

4. Page 17: Line 5 should make the definitions of the sets $\mathbf{S}_X$ and $\mathbf{S}'_X$ more explicit: let $\mathbf{S}_X$ be such that $X \in \mathbf{S}_X$ with $G[\mathbf{S}_X] \in C(G)$. Similarly for $\mathbf{S}'_X$, the line should read: let $\mathbf{S}'_X$ be such that $X \in \mathbf{S}'_X$ with $G'[\mathbf{S}'_X] \in C(G')$.

I

Identifying causal effects with the R package causaleffect

Tikka, S. and Karvanen, J.

Journal of Statistical Software, 76(12): 1–30, 2017.

Journal of Statistical Software

February 2017, Volume 76, Issue 12. *doi: 10.18637/jss.v076.i12*

Identifying Causal Effects with the R Package causaleffect

Santtu Tikka
University of Jyvaskyla

Juha Karvanen
University of Jyvaskyla

Abstract

Do-calculus is concerned with estimating the interventional distribution of an action from the observed joint probability distribution of the variables in a given causal structure. All identifiable causal effects can be derived using the rules of do-calculus, but the rules themselves do not give any direct indication whether the effect in question is identifiable or not. Shpitser and Pearl (2006b) constructed an algorithm for identifying joint interventional distributions in causal models, which contain unobserved variables and induce directed acyclic graphs. This algorithm can be seen as a repeated application of the rules of do-calculus and known properties of probabilities, and it ultimately either derives an expression for the causal distribution, or fails to identify the effect, in which case the effect is non-identifiable. In this paper, the R package **causaleffect** is presented, which provides an implementation of this algorithm. Functionality of **causaleffect** is also demonstrated through examples.

Keywords: DAG, do-calculus, causality, causal model, identifiability, graph, C-component, hedge, d-separation.

1. Introduction

When discussing causality, one often means the relationships between events, where a set of events directly or indirectly causes another set of events. The aim of causal inference is to draw conclusions from these relationships by using available data and prior knowledge. Causal inference can also be applied when determining the effects of actions on some variables of interest. These types of actions are often called interventions and the results of the interventions are referred to as causal effects.

The causal inference can be divided into three sub-areas: discovering the causal model from the data, identifying the causal effect when the causal structure is known and estimating an identifiable causal effect from the data. Our contribution belongs to the second category,

identification of causal effects. As a starting point, we assume that the causal relationships between the variables are known in a non-parametric form and formally presented as a probabilistic causal model (Pearl 1995). Part of the variables may be latent. The causal structure, i.e., the non-parametric causal relationships, can be described using a directed acyclic graph (DAG). A causal effect is called identifiable if it can be uniquely determined from the causal structure on basis of the observations only.

Do-calculus (Pearl 1995) consist of a set of inference rules, which can be used to express the interventional probability distribution using only observational distributions. The rules of do-calculus do not themselves indicate the order in which they should be applied. This problem is solved in the algorithm developed by Tian and Pearl (2003) and Shpitser and Pearl (2006b). The algorithm is proved to determine the interventional distribution of an identifiable causal effect. When faced with an unidentifiable effect, the algorithm provides a problematic graph structure called a hedge, which can be thought of as the cause of unidentifiability.

Other R packages for causal inference are summarized in Table 1. It can be seen that in addition to **causaleffect**, only **pcalg** (Kalisch *et al.* 2012) supports the identification of causal effects. **pcalg** supports the generalized back-door criterion but does not support the front-door criterion. Thus, according to our knowledge, **causaleffect** is the only R package that implements a complete algorithm for the identification of causal effects.

An algorithm equivalent to the one developed by (Shpitser and Pearl 2006b) has been implemented earlier by Lexin Liu in the **CIBN** software using **JavaBayes**, which is a graphical software interface written in Java by Fabio Gagliardi Cozman. In addition to causal effect identification **CIBN** also provides tools for creating and editing graphical models. **CIBN** is freely available from `http://web.cs.iastate.edu/~jtian/Software/CIBN.htm`. **DAGitty** (Textor *et al.* 2011) provides another free interface for causal inference and causal modeling. One of the main features of **DAGitty** is finding sufficient adjustment sets for the minimization of bias in causal effect estimation. **DAGitty** can also be used to determine instrumental variables, which is a feature currently not provided by **causaleffect**. However, **DAGitty** does not provide a complete criterion for identifiability.

Familiarity of Pearl's causal model, do-calculus and basic graph theory is assumed throughout the paper. These concepts are briefly reviewed in Appendix A. A more detailed description can be found in (Pearl 2009) and (Koller and Friedman 2009). Notation similar to that of (Shpitser and Pearl 2006b) is also utilized repeatedly in this paper. Capital letters denote variables and small letters denote their values. Bold letters denote sets which are formed of the previous two. The abbreviations $Pa(\mathbf{Y})_G$, $An(\mathbf{Y})_G$, and $De(\mathbf{Y})_G$ denote the set of observable parents, ancestors and descendants of the node set $\mathbf{Y}$ while also containing $\mathbf{Y}$ itself. It should also be noted that the shorthand notation of bidirected edges is used to represent the direct effects of an unobserved confounding variable on the two variables at the endpoints of the bidirected edge.

A motivating example is presented in Section 2. The identification algorithm is presented in Section 3 and the details of its R implementation are described in Section 4. Section 5 showcases the usage of **causaleffect** in R with some simple examples, and describes some curious special cases arising from the nature of the algorithm itself. Section 6 concludes this paper by providing some examples of similar algorithms, where the work of this paper could be applicable.

Packages for specific applications	
ASPBay	Bayesian inference on causal genetic variants using affected sib-pairs data (Dandine-Roulland 2015)
cin	Causal inference for neuroscience (Luo *et al.* 2011)
mwa	Causal inference in spatiotemporal event data (Schutte and Donnay 2015)
qtlnet	Causal inference of QTL networks (Neto and Yandell 2014)
Packages for estimation of causal effects from data	
CausalGAM	Estimation of causal effects with generalized additive models (Glynn and Quinn 2010)
InvariantCausalPrediction	Invariant causal prediction (Meinshausen 2016)
iWeigReg	Improved methods for causal inference and missing data problems (Tan and Shu 2013)
pcalg	Methods for graphical models and causal inference
SVMMatch	Causal effect estimation and diagnostics with support vector machines (Ratkovic 2015)
wfe	Weighted linear fixed effects regression models for causal inference (Kim and Imai 2014)
Packages for sensitivity analysis and other specific problems in causal inference	
causalsens	Selection bias approach to sensitivity analysis for causal effects (Blackwell 2015)
cit	Causal inference test (Millstein 2016)
ImpactIV	Identifying causal effect for multi-component intervention using instrumental variable method (Ding 2012)
inferference	Methods for causal inference with interference (Saul 2015)
MatchingFrontier	Computation of the balance – sample size frontier in matching methods for causal inference (King *et al.* 2015)
mediation	Causal mediation analysis (Tingley *et al.* 2014)
qualCI	Causal inference with qualitative and ordinal information on outcomes(Kashin *et al.* 2014)
SimpleTable	Bayesian inference and sensitivity analysis for causal effects from 2×2 and $2 \times 2 \times K$ tables in the presence of unmeasured confounding (Quinn 2012)
treatSens	Sensitivity analysis for causal inference (Carnegie *et al.* 2016)
Packages for causal discovery	
CAM	Causal additive model (CAM) (Peters and Ernest 2015)
D2C	Predicting causal direction from dependency features (Bontempi *et al.* 2015)
pcalg	Methods for graphical models and causal inference
Packages for identification of causal effects	
causaleffect	Deriving expressions of joint interventional distributions in causal models
pcalg	Methods for graphical models and causal inference

Table 1: R packages for causal inference.

2. Example on do-calculus

Consider identification of causal effect $P_x(y)$ in the graph G of Figure 1. We show how this causal effect can be identified by applying do-calculus (Pearl 2009) manually. Later the same example is reconsidered using the identification algorithm.

First, the rules of do-calculus are shortly reviewed. The purpose of do-calculus is to represent the interventional distribution $P_{\mathbf{x}}(\mathbf{y})$ by using only observational probabilities. A causal effect is identifiable, if such an expression can be found by applying the rules of do-calculus repeatedly. This result follow directly from the definition of identifiability due to the fact that all observational distributions are assumed identical for the causal models that induce G.

Let $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ be pairwise disjoint sets of nodes in the graph G induced by a causal model M. 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 $\mathbf{Z}$. Let P be the joint distribution of all observed and unobserved variables of M. Now, the following three rules hold (Pearl 1995):

1. Insertion and deletion of observations:

$$P_{\mathbf{x}}(\mathbf{y}|\mathbf{z},\mathbf{w}) = P_{\mathbf{x}}(\mathbf{y}|\mathbf{w}), \text{ if } (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}}}}.$$

2. Exchanging actions and observations:

$$P_{\mathbf{x},\mathbf{z}}(\mathbf{y}|\mathbf{w}) = P_{\mathbf{x}}(\mathbf{y}|\mathbf{z},\mathbf{w}), \text{ if } (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}},\underline{\mathbf{Z}}}}.$$

3. Insertion and deletion of actions:

$$P_{\mathbf{x},\mathbf{z}}(\mathbf{y}|\mathbf{w}) = P_{\mathbf{x}}(\mathbf{y}|\mathbf{w}), \text{ if } (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}},\overline{Z(\mathbf{W})}}},$$

   where $Z(\mathbf{W}) = \mathbf{Z} \setminus An(\mathbf{W})_{G_{\overline{\mathbf{X}}}}$.

The rules of do-calculus can be shown to be true by using d-separation and the definition of the $do(\cdot)$-operator. Pearl presented proofs for these three rules (Pearl 1995). Do-calculus has also been shown to be complete, meaning that the expressions of all identifiable causal effects can be derived by using the three rules (Shpitser and Pearl 2006b; Huang and Valtorta 2006).

To identify $P_x(y)$ in the causal model of Figure 1, we begin with the factorization

$$P_x(y) = \sum_{w,z} P_x(y|w,z)P_x(z|w)P_x(w). \tag{1}$$

Let us start by focusing on the first term in the sum. Because $(Y \perp\!\!\!\perp Z|X,W)_{G_{\overline{X},\underline{Z}}}$ rule 2 implies that

$$P_x(y|w,z) = P_{x,z}(y|w)$$

and by noting that $(Y \perp\!\!\!\perp X|Z,W)_{G_{\overline{Z},\overline{X}}}$ rule 3 allows us to write

$$P_{x,z}(y|w) = P_z(y|w).$$

By expanding the previous expression we get

$$P_z(y|w) = \sum_x P_z(y|w,x)P_z(x|w). \tag{2}$$

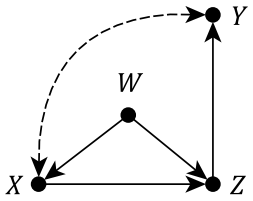


Figure 1: Graph G for the illustrative example.

Rule 2 and the fact that $(Y \perp\!\!\!\perp Z|X, W)_{G_{\underline{Z}}}$ together imply

$$P_z(y|w, x) = P(y|w, x, z). \tag{3}$$

The condition $(X \perp\!\!\!\perp Z|W)_{G_{\overline{Z}}}$ and rule 3 allow us to write

$$P_z(x|w) = P(x|w). \tag{4}$$

Inserting (3) and (4) into (2) yields

$$P_z(y|w) = \sum_x P(y|w, x, z)P(x|w). \tag{5}$$

Focusing now on the second term of (1) we see that because $(Z \perp\!\!\!\perp X|W)_{G_{\underline{X}}}$ rule 2 implies that

$$P_x(z|w) = P(z|x, w). \tag{6}$$

Similarly, the third term simplifies by using rule 3 and the condition $(W \perp\!\!\!\perp X)_{G_{\overline{X}}}$ rule 3.

$$P_x(w) = P(w). \tag{7}$$

Finally, we combine the results above by inserting (5), (6) and (7) into (1) which yields the expression for the causal effect.

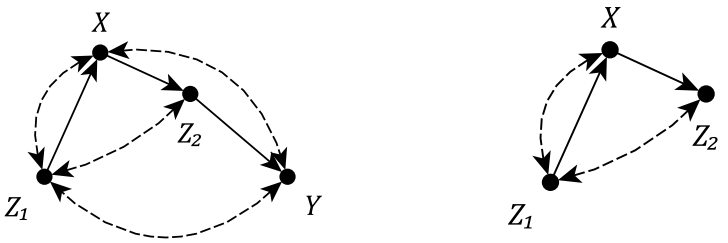
$$P_x(y) = \sum_{w,z} \left(\sum_x P(y|w, x, z)P(x|w) \right) P(z|x, w)P(w)$$

In Section 3.3 we will see how the causal effect can be identified by applying the algorithm of (Shpitser and Pearl 2006b). The previous result highly resembles the front-door criterion, which states that

$$P_{\mathbf{x}}(\mathbf{y}) = \sum_{\mathbf{s}} \left(\sum_{\mathbf{x}} P(\mathbf{y}|\mathbf{x}, \mathbf{s})P(\mathbf{x}) \right) P(\mathbf{s}|\mathbf{x}),$$

whenever the set $\mathbf{S}$ blocks all directed paths from $\mathbf{X}$ to $\mathbf{Y}$, there are no unblocked back-door paths from $\mathbf{X}$ to $\mathbf{S}$ and $\mathbf{X}$ blocks all back-door paths from $\mathbf{S}$ to $\mathbf{Y}$. However, neither W, Z, or $\{W, Z\}$ satisfy the role of the set $\mathbf{S}$. The criterion would certainly hold if we removed W from the graph.

3. Identifiability algorithm



(a) Graph G. (b) A subgraph of G induced by the set $\{X, Z_1, Z_2\}$.

Figure 2: An example illustrating the definition of an induced subgraph.

Even if a causal effect is identifiable, the rules of do-calculus themselves do not guarantee that they could be used to form an expression for the interventional distribution, and that it would contain only observed quantities. It is also not self-evident in which order the rules of do-calculus should be applied to reach the desired expression from the joint distribution of the observed variables $P(\mathbf{V})$.

To overcome these limitations an identifiability algorithm has been developed by Shpitser and Pearl (2006b). This algorithm can be used to determine the identifiability of any causal effect, in addition of generating the expression for the interventional distribution in the case of an identifiable effect.

3.1. Definitions

Some graph theoretic definitions are necessary in order to present the algorithm. The notation mostly follows that of (Shpitser and Pearl 2006b) with some slight alterations for the benefit of the reader.

Definition 1 (Induced Subgraph)**.** Let $H = \langle \mathbf{W}, \mathbf{F} \rangle$ and $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be graphs such that $\mathbf{W} \subset \mathbf{V}$. If every pair of nodes $X, Y \in \mathbf{W}$ is connected by an edge in graph H precisely when they are connected by an edge of the same direction in graph G, then H is an *induced subgraph* induced by the set $\mathbf{W}$ and $H = G[\mathbf{W}]$.

Defining new graphs using only a set of nodes can easily be achieved using induced subgraphs. For example, the graph in Figure 2(b) is an induced subgraph induced by the nodes X, Z_1 and Z_2 from G in 2(a).

Perhaps the most important definition is C-component (confounded component).

Definition 2 (C-component, (Shpitser and Pearl 2006b) 3)**.** Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be a graph. If there exists a set $\mathbf{B}$ such that $\mathbf{B} \subset \mathbf{E}$ and $\mathbf{B}$ contains only bidirected edges, and the graph $\langle \mathbf{V}, \mathbf{B} \rangle$ is connected, then G is a *C-component*.

Both graphs in Figure 2 are examples of C-components. Even if a graph is not a C-component, at least one of its subgraphs is guaranteed to be a C-component because every subgraph induced by a single node is always a C-component. It is often of greater interest to determine how a given graph can be partitioned in C-components that contain as many nodes as possible.

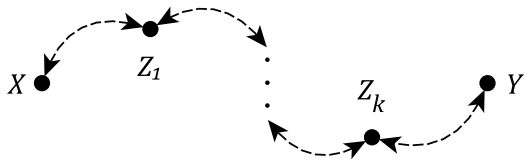


Figure 3: Path H.

Definition 3 (Maximal C-component)**.** Let G be a graph and $C = \langle \mathbf{V}, \mathbf{E} \rangle$ a C-component such that $C \subset G$. C-component C is *maximal* (with respect to graph G) if $H \subset C$ for every bidirected path H of graph G which contains at least one node of the set $\mathbf{V}$.

Tian (2002) proved, that the joint probability distribution $P(\mathbf{V})$ of the observed variables of graph G can always be factorized in such a way, that each term of the resulting product corresponds to a maximal C-component. This property is in a fundamental role in the algorithm, since it can be used to recursively divide the expression of the interventional distribution into simpler expressions.

If a given graph G is not a C-component, it can still be divided into a unique set $C(G)$ of subgraphs, each a maximal C-component of G. This follows from the fact, that there exists a bidirected path between two nodes in G if and only if they belong in the same maximal C-component, which in turn follows from the definition of a maximal C-component. This means, that the bidirected paths of graph G completely define its maximal C-components.

C-trees are a special case of C-components. They are closely related to direct effects, which are causal effects of the form $P_{Pa(Y)}(Y)$.

Definition 4 (C-tree, (Shpitser and Pearl 2006b) 4)**.** Let G be a C-component such that every observed node has at most one child. If there is a node Y such that $G[An(Y)_G] = G$, then G is a *Y-rooted C-tree*.

Using only C-trees and C-components it is already possible to characterize identifiability of effects on a single variable. C-forest is the multivariate generalization of a C-tree in such a way that the *root set*, which is the set of nodes $\{X \in G \mid De(X)_G \setminus \{X\} = \emptyset\}$, contains one or more nodes.

Definition 5 (C-forest, (Shpitser and Pearl 2006b) 5)**.** Let G be a graph and $\mathbf{Y}$ its root set. If G is a C-component, and every observed node has at most one child, then G is *$\mathbf{Y}$-rooted C-forest*.

Both C-components in Figure 2 are also C-forests, because every observed node has at most one child in both graphs. In addition, their root sets consist only of a single node. There exists a connection between C-forests and general causal effects of the form $P_{\mathbf{x}}(\mathbf{Y})$. A graph structure formed by a pair of C-trees is used to determine such effects.

Shpitser and Pearl (2006b) proved, that if a graph G contains a hedge for $P_{\mathbf{x}}(\mathbf{y})$, then the effect is not identifiable.

Definition 6 (Hedge, (Shpitser and Pearl 2006b) 6)**.** Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be a graph, and $\mathbf{X}, \mathbf{Y} \subset \mathbf{V}$ disjoint subsets. If there are two $\mathbf{R}$-rooted C-forests $F = \langle \mathbf{V}_F, \mathbf{E}_F \rangle$ and $F' = \langle \mathbf{V}_{F'}, \mathbf{E}_{F'} \rangle$ such that $\mathbf{V}_F \cap \mathbf{X} \neq \emptyset$, $\mathbf{V}_{F'} \cap \mathbf{X} = \emptyset$, $F' \subset F$, and $\mathbf{R} \subset An(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$, then F and F' form *hedge* for $P_{\mathbf{x}}(\mathbf{y})$ in G.

Hedges are a remarkable structure, since they generalize certain results regarding identifiability. One example of such a result is the condition for identification of a causal effect of the form $P_x(\mathbf{y})$ in (Tian and Pearl 2002). The result states that $P_x(\mathbf{y})$ is identifiable if and only if there are no bidirected paths between X and any of its children in $G[An(\mathbf{Y})_G]$. Consider the graph $H = \langle \mathbf{V}, \mathbf{E} \rangle$ in Figure 3 containing the nodes X and Y and a bidirected path connecting them formed by the intermediary nodes $\{Z_1, \ldots, Z_k\}$. One can observe, that the C-forests H and $H[\mathbf{V} \setminus \{X\}]$ form a hedge for $P_x(Y, Z_1, \ldots, Z_k)$.

3.2. Algorithm

Using the previously presented definitions it is now possible to define Algorithm 1, which completely characterizes the identifiability problem of general causal effects. Shpitser and Pearl (2006b) showed, that the expression returned by Algorithm 1 for $P_{\mathbf{x}}(\mathbf{y})$ is always correct if the effect in question is identifiable. They also showed, that if the algorithm is interrupted on line five, then the original graph G contains a hedge, preventing the identifiability of the effect. The existence of a hedge is therefore equivalent with unidentifiability. This result also shows the completeness of do-calculus, because the algorithm only applies standard rules of probability manipulations and the three rules of do-calculus. All variables are assumed to be discrete, but the algorithm can also be applied in a continuous case, when the respective sums are replaced with integrals.

The algorithm is required to be able to iteratively process the nodes of the graph, which means that the nodes have to be ordered in some meaningful fashion. This ordering must be able to take the directions of the edges into account, and at least one such ordering must always exist for any given graph. Topological ordering has all of these prerequisite properties.

Definition 7 (Topological Ordering)**.** *Topological ordering* π of a DAG $G = \langle \mathbf{V}, \mathbf{E} \rangle$ is an ordering of its nodes, where either $X > Y$ or $Y > X$ for all pairs of nodes $X, Y \in \mathbf{V}$, $X \neq Y$ in G. In addition, no node can be greater than its descendants in π. In other words, if X is an ancestor of Y in G, then $X < Y$.

There exists at least one topological ordering for any DAG, but in some cases there can be multiple orderings. One way to always construct an ordering for a given graph is to begin by determining all nodes without parents, and ordering them arbitrarily. Next, all nodes without parents excluding the nodes found in previous step are determined and again ordered arbitrarily. It is also assigned, that the largest node in the previous step is smaller than the smallest node in the current step. This process is iterated, until all nodes have been ordered.

Algorithm 1 is simple in a sense that at each recursion stage the computation proceeds to exactly one line only. This is easy to see from the fact that after a condition regarding any of the line has been checked, either a **return** or a **FAIL** command will be executed. If $\mathbf{x} = \emptyset$ on line one, then the marginal distribution $P(\mathbf{y})$ is computed instead of a causal effect. This can be achieved by marginalizing over the joint distribution $P(\mathbf{V})$. On line two, all non-ancestors of $\mathbf{Y}$ in G are eliminated. This is possible due to the fact that the input of the algorithm assumes that G is an I-map of G and thus all necessary conditional independences hold. On line three, interventions are added to the original causal effect, which is feasible due to the third rule of do-calculus, because $(\mathbf{Y} \perp\!\!\!\perp \mathbf{W} | \mathbf{X})_{G_{\overline{\mathbf{X}},\overline{\mathbf{W}}}}$.

It is possible to index the nodes of G and the nodes of any subgraph of G using the topological ordering. This property is utilized on lines four, six and seven. The notation $V_\pi^{(i-1)}$ refers

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 **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 $\sum_{v \in \mathbf{s} \setminus \mathbf{y}} \prod_{V_i \in \mathbf{S}} P(v_i | v_\pi^{(i-1)})$.
7: **if** $(\exists \mathbf{S}') \mathbf{S} \subset \mathbf{S}'$ such that $G[\mathbf{S}'] \in C(G)$, **then**
return ID$(\mathbf{y}, \mathbf{x} \cap \mathbf{s}', \prod_{V_i \in \mathbf{S}'} P(V_i | V_\pi^{(i-1)} \cap \mathbf{S}', v_\pi^{(i-1)} \setminus \mathbf{s}'), G[\mathbf{S}'])$.

Algorithm 1: The causal effect of intervention $do(\mathbf{X} = \mathbf{x})$ on $\mathbf{Y}$.

to all nodes in G that are smaller than V_i in π. Any topological ordering of G is also a topological ordering for any subgraph of G. This means, that it is unnecessary to determine a new ordering for each subgraph of G. Instead, one can fix the ordering before applying the algorithm.

The maximal C-components of $G[\mathbf{V} \setminus \mathbf{X}]$ are determined on line four and their factorization property is utilized. If more than one C-components were found, it is now necessary to calculate a new causal effect for every C-component. The algorithm proceeds to either line five, six or seven in the case if only one C-component was found.

If Algorithm 1 throws **FAIL**, then the original graph G contains a hedge formed by graph G and $G[\mathbf{S}]$ of the current recursion stage, due to which the original effect is not identifiable and computation terminates. If the algorithm continues, then it is necessary to determine whether $G[\mathbf{S}]$ is a maximal C-component of G. If this is the case, then the condition of line six has been satisfied. In the other case, the computation of the intervention can be limited to the intersection of sets $\mathbf{X}$ and $\mathbf{S}'$ on line seven.

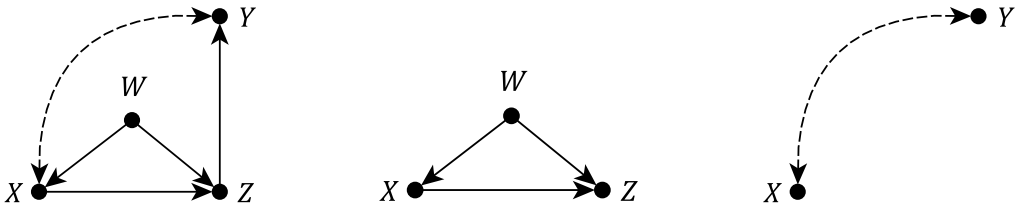
Identifiability of conditional interventional distributions is characterized by Algorithm 2. This algorithm is a generalization of Algorithm 1 and in fact it utilizes the function **ID** in the computation. It was constructed by Shpitser and Pearl (2006a) for identifying conditional causal effects i.e., causal effects of the form $P_{\mathbf{x}}(\mathbf{y}|\mathbf{z})$. They showed, that this algorithm is also sound and complete for identifying all such effects.

The primary focus of this paper however, is the implementation of Algorithm 1. The imple-

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 **FAIL**(F, F').

function IDC$(\mathbf{y}, \mathbf{x}, \mathbf{z}, P, G)$
1: **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$(\mathbf{y}, \mathbf{x} \cup \{z\}, \mathbf{z} \setminus \{z\}, P, G)$.
2: **else let** $P' = \mathbf{ID}(\mathbf{y} \cup \mathbf{z}, \mathbf{x}, P, G)$.
return $P' / \sum_{y \in \mathbf{y}} P'$

Algorithm 2: The causal effect of intervention $do(\mathbf{X} = \mathbf{x})$ on $\mathbf{Y}$ given $\mathbf{Z}$.



(a) Graph G (b) Subgraph $G[An(Z)_G]$. (c) Subgraph $G[\mathbf{S}']$.

Figure 4: Graph G and its subgraphs.

mentation of Algorithm 2 follows seamlessly from this implementation, because at the bottom of any recursive stack of **IDC** the function **ID** is ultimately called, which determines if the original conditional effect is identifiable. The only additional task is to determine whether a suitable node for the d-separation condition exists on line 1.

3.3. Application in practice

We return to the example presented in Section 2. The graph of the example along with some subgraphs are shown here in Figure 4. Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be a graph such as in Figure 4(a) and a causal effect of interest $P_x(y)$, which is to be identified from the joint distribution $P(X, Y, Z, W)$. Only a single topological ordering exists for the nodes of G, and it is $W < X < Z < Y$. Clearly $\mathbf{x} \neq \emptyset, \mathbf{V} = An(Y)_G$ and $\mathbf{W} = \emptyset$, so the first three lines are ignored and line four is triggered, since

$$C(G[\mathbf{V} \setminus \{X\}]) = \{G[W], G[Z], G[Y]\}.$$

Because $\mathbf{v} \setminus (\{y\} \cup \{x\}) = \{w, z\}$, it is now necessary to identify three new causal effects in the following expression:

$$\sum_{w,z} P_{x,z,y}(w) P_{w,x,y}(z) P_{w,x,z}(y).$$

Consider the first term of the product. Because $\mathbf{V} \neq An(W)_G$, line two is triggered, and non-ancestors of W are ignored. This results in the first term simplifying to $P(w)$ because

$An(W)_G = \{W\}$. Line two is also triggered when computing the second term, and

$$P_{w,x,y}(z) = P_{w,x}(z)$$

in a subgraph induced by ancestors of Z as in Figure 4(b). Observing that

$$C(G[An(Z)_G \setminus \{W, X\}]) = \{G[Z]\}$$

and

$$G[Z] \in C(G[An(Z)_G]) = \{G[X], G[W], G[Z]\},$$

the algorithm proceeds to line 6 and the second term simplifies again

$$P_{w,x}(z) = P(z|w, x).$$

The last term $P_{w,x,z}(y)$ triggers line four, because

$$C(G[\mathbf{V} \setminus \{W, X, Z\}]) = \{G[Y]\}.$$

$G[Y]$ is not a maximal C-component of G, but Y is a node of one of the maximal C-components of G: $\{Y\} \subset \{X, Y\} = \mathbf{S}'$. It holds for the set $\mathbf{S}'$, that

$$G[\mathbf{S}'] \in C(G) = \{G[\{X, Y\}], G[W], G[Z]\}.$$

So it is mandatory to compute $P_x(y)$ from $P(X|w)P(Y|X, w, z)$ in the graph corresponding to Figure 4(c). It should be noted, that this causal effect differs from the original effect $P_x(y)$, because the joint distribution $P(\mathbf{V})$ of observed variables of G is not the same as the distribution $P(X|w)P(Y|X, w, z)$ of the subgraph of the current recursion stage.

Line two is triggered next, and since Y has no observed ancestors in the graph corresponding to 4(c), it follows that

$$P_x(y) = \sum_x P(x|w)P(y|x, w, z).$$

An expression for the original causal effect is obtained by combining the previous results

$$P_x(y) = \sum_{w,z} P(z|w, x)P(w) \sum_x P(y|w, x, z)P(x|w).$$

The result agrees with the result derived in Section 2.

Algorithm 1 can also be used to detect unidentifiability. Let $F = \langle \mathbf{V}, \mathbf{E} \rangle$ be a graph of Figure 5(a) and a causal effect of interest $P_x(y)$, which is to be identified from $P(X, Y, Z_1, Z_2)$. Let the topological ordering of the nodes of F be $Z_1 < X < Z_2 < Y$.

The computation starts from line three

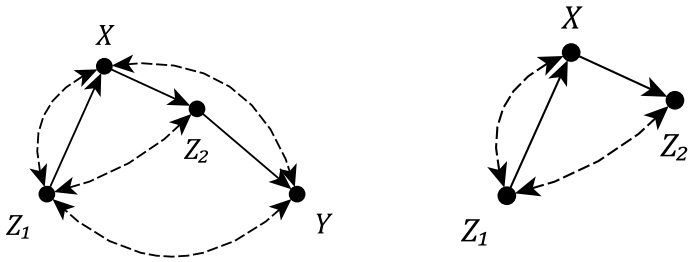
$$\mathbf{W} = (\mathbf{V} \setminus \mathbf{X}) \setminus An(\mathbf{Y})_{F_{\overline{\mathbf{X}}}} = (\{X, Y, Z_1, Z_2\} \setminus \{X\}) \setminus \{X, Z_2, Y\} = \{Z_1\} \neq \emptyset.$$

Z_1 is added to the original intervention, so $P_{z_1,x}(y)$ has to be identified. Line four is triggered next, because

$$C(F[\mathbf{V} \setminus \{Z_1, X\}]) = \{F[Z_2], F[Y]\}.$$

Since $\mathbf{v} \setminus (\{y\} \cup \{z_1, x\}) = \{z_2\}$, two new causal effects have to be identified following expression:

$$\sum_{z_2} P_{z_1,x,y}(z_2)P_{z_1,x,z_2}(y).$$



(a) Graph F. (b) Subgraph $An(Z_2)_F$.

Figure 5: Graph F and its subgraph $F[An(Z_2)_F]$.

Consider the first term of the product. Clearly $\mathbf{V} \neq An(Z_2)_F$, so the algorithm proceeds to line two, which means that

$$P_{z_1,x,y}(z_2) = P_{z_1,x}(z_2)$$

in a subgraph formed by ancestors of Z_2 as in Figure 5(b). However, $P_{z_1,x}(z_2)$ is not identifiable, because

$$C(F[An(Z_2)_F \setminus \{Z_1, X\}]) = \{F[Z_2]\} \text{ and } C(F[An(Z_2)_F]) = \{F[An(Z_2)_F]\},$$

which trigger line five. In conclusion, F contains a hedge for $P_{z_1,x}(z_2)$ formed by C-forests $F[Z_2]$ and $F[\{Z_1, Z_2, X\}]$. Thus the original effect $P_x(y)$ is not identifiable.

4. Implementation using R

The programming language R (R Core Team 2016) was chosen for the implementation of Algorithm 1. The R packages **XML** (Temple Lang 2016), **igraph** (Csardi and Nepusz 2006) and **ggm** (Marchetti *et al.* 2015) are utilized repeatedly throughout the implementation.

4.1. Graph files

A graph G induced by the causal model is a crucial argument of Algorithm 1. Many file formats for visualizing graphs are available, each with their own strengths and weaknesses. Some of these formats are very simple, and do not differentiate directed and undirected graphs. Some formats offer excessive features for describing causal models, or they might require handling complex syntax, which can be time consuming.

GraphML (Brandes *et al.* 2002) is a user-friendly file format for graphs. Its features include support for directed graphs and visualizations. GraphML is based on the extensible markup language XML (Maler *et al.* 2004), which makes processing of graphs files almost effortless. One can also include the names of the nodes within the GraphML file itself, so the user is not limited to having to input the node names themselves inside the R environment. Graphical editors for creating GraphML files are freely available for the user. A special function called `parse.graphml` has been developed for processing GraphML files. However, the implementation of Algorithm 1 is not limited to GraphML files alone. Any file format supported by the **igraph** package can be used, as long as the graph follows one of the following notations for bidirected edges.

(a) Notation 1. (b) Notation 2. (c) Notation 3.

Figure 6: Notations for bidirected edges.

Bidirected edges can be separated from unidirected edges by using graphical parameters. For this purpose, three distinct notations have been selected to describe bidirected edges, which correspond to unobserved nodes.

The available notations for bidirected edges are shown in Figures 6(a), 6(b) and 6(c). It should be noted, that notations 1 and 2 are almost identical. Because of their similarity, both notations 1 and 2 are referred to as `standard` notation. Notation 3, as shown in Figure 6(c) differs from the previous two. It is apparent, that this notation cannot be used as such, because it induces loops in the graph which is not allowed in the context of DAGs. However, GraphML format enables the assignment of parameters for the edges, which in turn allows one to separate these edges from their unidirected counterparts. When using notation 3, one must define a parameter called `description` for the two unidirected edges corresponding to the bidirected edge, and assign its value to `"U"` (Unobserved). Notation 3 is used in the implementation itself, which is why it is referred to as the `internal` notation.

The process of importing GraphML files created by a graphical editor is handled by using the R package **XML**. This package contains the function `xmlParse`, which is utilized to import graph files into R objects. It should be noted, that these objects only reflect their internal C objects and are thus different from ordinary R objects. This means that the memory reserved by the XML objects has to be freed after the files have been imported. Normally R does this automatically.

Algorithm 1 requires only a small portion of the XML content, and the unnecessary content is removed in the process of searching for the important items. Items of importance are those that contain data about the node names, node count, edge count and the values of the `description` parameters of the edges. If notation 1 or 2 of Figures 6(a) and 6(b) was used for the bidirected arcs, it is converted to match the `internal` format of Figure 6(c). The XML search is implemented using the function `getNodeSet` of the **XML** package. This function uses XPath, which is a processing language for XML content search (Simpson 2002).

When the crucial information has been extracted, an **igraph** graph is formed from the remaining content. **igraph** is a tool for visualizing and processing graphs, and it can handle graphs which may contain millions of nodes due to its implementation in C. This package also offers many useful functions related to Algorithm 1, such as determining the ancestors of a node, constructing a topological ordering and generating induced subgraphs from a set of edges or nodes. One of the main goals of **igraph** is the effortless implementation of graph algorithms.

4.2. Distribution objects

An important question regarding the Algorithm 1 of Section 3.2, is how the probability distribution which changes at each recursive stage should be implemented. An intuitive solution is to construct a distribution object, which maintains the terms currently present in

the expression. Distribution objects are recursive by construction as is the algorithm itself. In practice this means that when any of the lines four, six or seven is triggered, sub objects are formed, which correspond to the product terms of the expression. These sub objects can further branch into sub objects of their own and so forth. **causaleffect** implements an R class called `probability` to represent the distribution objects.

Multiple attributes have to be set for the distribution objects in order to present the probability distribution precisely. The string vectors `var` and `cond` are one of the most common attributes, because they enable the definition of a simple conditional distribution. A distribution is formed by the variables described in `var` conditioned on those of `cond`. For example, let `p` be a distribution object, and let the values of its attributes be `var = "Y"` and `cond = "X"`. Therefore object `p` represents the conditional distribution $P(Y|X)$.

When the distribution is a product, the individual terms are defined in a list of distribution objects called `children` and a logical variable `recursive` is set to `TRUE` to differentiate this object from those containing only a single term. For example, for a distribution which represents the distribution $P^* = P(Z|X)P(X|Y)P(Y)$ one has to set `children = list(a,b,c)`, where the objects `a`, `b` and `c` represent the distributions $P(Z|X), P(X|Y)$ and $P(Y)$ respectively.

For marginal distributions a string vector `sumset` has been defined. The contents of this vector correspond to the variables which the distribution is to be summed over in the discrete case, or integrated over in the continuous case. In simple situations this parameter is not needed, but often with more complex graphs one encounters instances, where the computation of conditionals is no longer straightorward. Suppose one had to compute the marginal distribution $P^*(X)$ of X from the joint distribution $P^*(X,Y,Z)$ of the previous example. To achieve this, one has to set `sumset = c("Y","Z")` for the matching distribution object, because $P^*(X) = \sum_{Y,Z} P(Z|X)P(X|Y)P(Y)$.

The level of complexity increases further when computing conditionals from distributions which consist of multiple product terms. The previously presented attributes are often insufficient to form an expression for the corresponding distribution object. Consider once more the joint distribution P^*. Computing the marginal conditional distribution $P^*(X|Y)$ results in

$$P^*(X|Y) = \frac{P^*(X,Y)}{P^*(Y)} = \frac{\sum_Z P(Z|X)P(X|Y)P(Y)}{\sum_{X,Z} P(Z|X)P(X|Y)P(Y)} = \frac{P(X|Y)\sum_Z P(Z|X)}{\sum_X P(X|Y)\sum_Z P(Z|X)} = P(X|Y).$$

The implementation is able to handle similar situations, where the expression can easily be simplified using the following procedure. Any term which does not depend on the summation index, will be placed outside of the sum. Next, it is checked whether any expressions can be simplified by changing the order of summation. Corresponding terms are subtracted if possible.

These simplification rules are not sufficient to handle every situation. For example, the expression $\sum_X P(Y|X)P(X)$ cannot be simplified using the procedure above. One cannot remove any terms from within the sum and the summation order is clearly fixed. In situations, where the denominator is necessary in order to correctly form the expression, one needs to include additional attributes called `divisor` and `fraction`. These attributes are similar to the attributes `children` and `recursive` in a sense that `divisor` contains the distribution

object that represents the denominator and `fraction` is set to `TRUE` when it is necessary to represent the expression as a fraction.

4.3. Maximal C-components

In Section 3.1 it was shown, that for every causal diagram G there exists a unique set $C(G)$ of maximal C-components of G. To construct this set, one has to begin by determining all bidirected edges of G. Afterwards, a subgraph containing only bidirected edges is formed. This subgraph will contain one or more *components*, which are connected subgraphs of G. Because these components are disjoint and every pair of nodes within a component is connected by a bidirected path, it follows that they must be the maximal C-components of G. The *adjacency matrix* of G is utilized to find the bidirected edges of G.

Definition 8 (adjacency matrix)**.** An *adjacency matrix* of a graph $G = \langle \mathbf{V}, \mathbf{E} \rangle$ is a $n \times n$ matrix $A = [a_{ij}]$, where n is the number of nodes of G, $\mathbf{V} = \{V_1, V_2, \ldots, V_n\}$ and a_{ij} is the number of edges from V_i to V_j.

Because G is directed, its adjacency matrix is not necessarily symmetric. When notation 3 of Figure 6(c) is used to describe the bidirected edges, it is easy to confirm that two nodes V_i and V_j are connected by at least one bidirected edge if and only if $a_{ij} \geq 1$ and $a_{ji} \geq 1$. Thus all bidirected edges can be determined by comparing A to its transpose $A^\top$, and by choosing only those edges which correspond to indices with $a_{ij} \geq 1$ and $a_{ji} \geq 1$.

The subgraph of G containing only bidirected edges is constructed by using the function `subgraph.edges` of the **igraph** package. This function retains all nodes of the input graph, but removes all the edges that were not given as input. The subgraph returned by this function is further divided into components by using the function `decompose.graph` which is also provided by **igraph**.

4.4. Implementation

All necessary preparations have been presented to implement Algorithm 1. Any probability distribution can be represented with a corresponding distribution object, and the adjacency matrix provides a method to determine the maximal C-components of G. Other important methods are provided by the **igraph** package, such as constructing subgraphs and determining the ancestors of a given set of nodes. In this implementation, the input of Algorithm 1 consists of the sets `x` and `y` including the graph G, and returns a `probability` object, which is a list structure that describes the expression of the causal distribution $P_{\mathbf{x}}(\mathbf{y})$ in terms of $P(\mathbf{V})$. The returned object can be further parsed into a character representation.

The R function of Algorithm 1 is called `id`. This function takes five parameters as input: a string vector `y`, a string vector `x`, a distribution object `P`, an **igraph** graph `G` and a string vector `to`. The first four parameters correspond to their mathematical counterparts, namely the vectors `x`, `y`, P and G. The last parameter `to` is a string vector representing some topological ordering of the nodes of G. All required set theoretic operations are included in R as the functions `intersect`, `setdiff` and `union`.

The observed portion of `G` is saved as `G.obs`. This graph contains all the observed nodes of G and the edges between them. In addition, the observed nodes are saved into vector `v`, and the ancestors of `y` are saved into vector `anc`. The implementation of each line of Algorithm 1 is presented next.

1: **if** $\mathbf{x} = \emptyset$, **then**
return $\sum_{v \in \mathbf{v} \setminus \mathbf{y}} P(\mathbf{v})$.

The truth value of the expression $\mathbf{x} = \emptyset$ is determined on line 1. This is done by computing the length of `x`. If the length is zero, then `id` combines the difference of the sets `v` and `y` with the `sumset` of `P` and returns `P`.

2: **if** $\mathbf{V} \neq An(\mathbf{Y})_G$, **then**
return $\mathbf{ID}(\mathbf{y}, \mathbf{x} \cap An(\mathbf{Y})_G, P(An(\mathbf{Y})_G), G[An(\mathbf{Y})_G])$.

The truth value of the condition on line 2 is determined by computing the length of the vector `setdiff(v, anc)`. If the length is not zero, then `id` is called with the arguments `id(y, intersect(x, anc), P, anc.graph, to)`, where `anc.graph` is the induced subgraph $G[An(\mathbf{Y})_G]$, which is constructed by using the `induced.subgraph` function of the **igraph** package. This function takes a set of nodes and a graph as input, and constructs a subgraph, which retains all of the nodes given as input, and all of the edges between them in the original graph.

3: **let** $\mathbf{W} = (\mathbf{V} \setminus \mathbf{X}) \setminus An(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$.
if $\mathbf{W} \neq \emptyset$, **then**
return $\mathbf{ID}(\mathbf{y}, \mathbf{x} \cup \mathbf{w}, P, G)$.

To construct a vector `w` which represents the node set $\mathbf{W}$, one must first construct the subgraph $G_{\overline{\mathbf{X}}}$. To accomplish this, all incoming edges of $\mathbf{X}$ have to be determined. A useful operator is provided by the **igraph** package to accomplish this. The operator `%->%` can be used to find incoming or outgoing edges of a node. In this case, one finds the incoming nodes of `x` with the command `E(G) [1:length(E(G)) %->% x]`, where `E` is a function that returns all edges of `G`. When the subgraph has been constructed, `w` can also be constructed. If the length of `w` is not zero, then `id` is called with the arguments `id(y, union(x, w), P, G, to)`.

4: **if** $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}_1], \ldots, G[\mathbf{S}_k]\}$, **then**
return $\sum_{v \in \mathbf{v} \setminus (\mathbf{y} \cup \mathbf{x})} \prod_{i=1}^{k} \mathbf{ID}(\mathbf{s}_i, \mathbf{v} \setminus \mathbf{s}_i, P, G)$.

The set $C(G[\mathbf{V} \setminus \mathbf{X}])$ can be found with the function `c.components`. This function determines the node set of every maximal C-component of the input graph, and returns them as a list `s`. If the length of this list is larger than one, then `id` returns a new distribution object with `sumset = setdiff(v, union(y, x)), recursive = TRUE, children = productlist`, where every object in `productlist` is determined by a new recursive call for every C-component $G[\mathbf{S}_i]$, $i = 1, \ldots, k$ that was found. These components are constructed by calling `id` with the arguments `id(s[[i]], setdiff(v, s[[i]]), P, G, to)`, $i = 1, \ldots, k$.

If the algorithm did not proceed to any of the previous lines, then the additional condition $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}]\}$ must be true. The node set of the single C-component $G[\mathbf{S}]$ is now saved in the vector `s`, which was previously a list. This means that `s` is replaced by `s[[1]]`.

5: **if** $C(G) = \{G\}$, **then**
throw $\mathbf{FAIL}(G, G[\mathbf{S}])$.

The function `c.components` is utilized again in order to find the maximal C-components of G. If in addition to having only a single C-component this C-component is G itself, then line five is triggered. This is checked by comparing `s` and `v`. If they are equal, then the computation is interrupted by the `stop` function and an error message is produced. The error message

describes the C-forests which form the problematic hedge structure for the causal effect of the current recursion stage.

6: **if** $G[\mathbf{S}] \in C(G)$, **then**
return $\sum_{v \in \mathbf{s} \setminus \mathbf{y}} \prod_{V_i \in \mathbf{S}} P(v_i | v_\pi^{(i-1)})$.

If the single C-component found on line four is one of the maximal C-components of G, then the function `id` returns a new distribution object. The `sumset` of this object is set to `setdiff(s, y)`. The distribution is a product so it must also be set, that `recursive = TRUE` for this new object. The objects in the list `children` are determined by new recursive calls for every node V_i in $\mathbf{S}$. The conditioning nodes are the ones that precede V_i in the topological ordering `to`.

7: **if** $(\exists \mathbf{S}')\mathbf{S} \subset \mathbf{S}'$ such that $G[\mathbf{S}'] \in C(G)$, **then**
return ID$(\mathbf{y}, \mathbf{x} \cap \mathbf{s}', \prod_{V_i \in \mathbf{S}'} P(V_i | V_\pi^{(i-1)} \cap \mathbf{S}', v_\pi^{(i-1)} \setminus \mathbf{s}'), G[\mathbf{S}'])$.

If the single C-component found on line four is not one of the maximal C-components of G, then it must be a subgraph of some maximal C-component $G[\mathbf{S}']$. Vector `s` is replaced by a vector corresponding to the nodes of $\mathbf{S}'$, since the nodes of $\mathbf{S}$ are no longer required. The function `id` is called with the following attributes `id(y, intersect(x, s), probability(recursive = TRUE, children = productlist), s.graph, to)`, where `s.graph` is the induced subgraph $G[\mathbf{S}']$ and every distribution object in `productlist` is constructed by setting `var <- s[i]` and `cond <- v[0:(ind[i]-1)]` for every node V_i in $\mathbf{S}'$.

Algorithm 2 is also implemented in **causaleffect** as the function `idc`. This function iterates through the nodes `z` which it receives as input in addition to the parameters that were previously defined for the `id` function. The d-separation condition on line 1 is checked by using the function `dSep` from the **ggm** package.

5. Package causaleffect

The primary goal of the **causaleffect** package is to provide the implementation described in Section 4. The package also provides a means of importing GraphML files into R while retaining any attributes that have been set for the nodes or edges of the graph.

5.1. Using causaleffect in R

The primary function which serves as a wrapper for the functions `id` and `idc` is called `causal.effect`. This function can be called as

```
causal.effect(y, x, z = NULL, G, expr = TRUE)
```

where the parameters `y`, `x` and `G` are identical to those of `id`. The parameter `z` is optional and it is used to represent the conditioning variables of `idc`. The initial probability object `P` which is a parameter of `id` does not have to be specified by the user. In essence, `causal.effect` starts from an empty distribution object, and gradually builds the final expression if possible. Also, the topological ordering `to` of the function `id` is automatically generated by the `topological.sort` function of the **igraph** package. It is verified, that the vectors `y`, `x` and `z` actually contain nodes that are present in `G`. If `G` is not a DAG then `causal.effect` will also terminate. The last parameter `expr` is a logical variable. If assigned to `TRUE`, `causal.effect`

will return the expression in LaTeX syntax. Otherwise, the `probability` object used internally by `id` is returned, which can be manually parsed by the user to gain the desired output. The function `get.expression` is also provided to get a string representation of a `probability` object. This function currently supports LaTeX syntax only.

First, **causaleffect** is loaded to demonstrate the usage of the package.

```
R> library("causaleffect")
```

The `causal.effect` function can be utilized without first importing a graph file. One can utilize the **igraph** package to construct graphs within R itself. This is demonstrated by replicating some of the graphs of Section 3.3. The graph of Figure 1 is created as follows.

```
R> library("igraph")
R> fig1 <- graph.formula(W -+ X, W -+ Z, X -+ Z, Z -+ Y, X -+ Y, Y -+ X,
+    simplify = FALSE)
R> fig1 <- set.edge.attribute(graph = fig1, name = "description",
+    index = c(5,6), value = "U")
R> ce1 <- causal.effect(y = "Y", x = "X", z = NULL, G = fig1, expr = TRUE)
R> ce1

[1] "\\left(\\sum_{W,Z}P(W)P(Z|W,X)\\left(\\sum_{X}P(Y|W,X,Z)P(X|W)\\right)\\right)"
```

Here `X -+ Z` denotes a directed edge from X to Z. The argument `simplify = FALSE` allows the insertion of duplicate edges for the purposes of forming bidirected arcs. Recalling the `internal` notation from Section 4.1 we must denote the unidirected edges that correspond to a bidirected edge with a special `description` parameter, and assign its value to `"U"`. This can be done with the `set.edge.attribute` function of the **igraph** package. Finally, the expression for the interventional distribution is obtained by using the `causal.effect` function. Usually one needs to apply the standard R function `cat` to obtain the expression with only singular slash symbols.

```
R> cat(ce1)

\left(\sum_{W,Z}P(W)P(Z|W,X)\left(\sum_{X}P(Y|W,X,Z)P(X|W)\right)\right)
```

To observe unidentifiability, the graph of Figure 5(a) is also constructed and an attempt is made to identify $P_x(y)$.

```
R> fig5 <- graph.formula(Z_1 -+ X, X -+ Z_2, Z_2 -+ Y, Z_1 -+ X, X -+ Z_1,
+    Z_1 -+ Z_2, Z_2 -+ Z_1, Z_1 -+ Y, Y -+ Z_1, X -+ Y, Y -+ X,
+    simplify = FALSE)
R> fig5 <- set.edge.attribute(graph = fig5, name = "description",
+    index = 4:11, value = "U")
R> causal.effect(y = "Y", x = "X", z = NULL, G = fig5, expr = TRUE)

Error: Graph contains a hedge formed by C-forests of nodes:
  {Z_1,X,Z_2} and {Z_2}.
```

The identification fails in this case due to a hedge present in the graph.

Another function provided by **causaleffect** is `parse.graphml` which can be called as

```
parse.graphml(file, format = c("standard", "internal"), nodes = c(),
  use.names = TRUE)
```

Parameter `file` is the path to the GraphML file the user wishes to convert into an **igraph** graph. Parameter `format` should match the notation that is used to denote bidirected edges in the input graph. The vector `nodes` can be used to give names to the nodes of the graph if they have not been specified in the file itself or alternatively, to replace them. Finally, `use.names` is a logical vector indicating whether the names of the nodes should be read from the file or not.

We provide an example GraphML file in the replication materials to demonstrate the use of the `parse.graphml` function. The file `g1.graphml` contains the graph of Figure 1 in `standard` notation. This means that we do not have to provide names for the nodes or set the unidentified edges manually. First, we read the file into R. This produces several warnings which can be ignored because they are related to the visual attributes created by the graphical editor that was used to produce `g1.graphml`. These attributes play no role in the identification of $P_x(y)$. We omit these warnings from the code for clarity.

```
R> gml1 <- parse.graphml("g1.graphml", format = "standard")
R> ce2 <- causal.effect(y = "Y", x = "X", z = NULL, G = gml1, expr = TRUE)
R> cat(ce2)

\left(\sum_{W,Z}P(W)P(Z|W,X)\left(\sum_{X}P(Y|W,X,Z)P(X|W)\right)\right)
```

We see that the result agrees with the one derived from the manually constructed graph.

For conditional causal effects, we simply utilize the parameter `z` of the `causal.effect` function. For example, we can obtain the formula for $P_x(z|w)$ in the graph of Figure 1.

```
R> cond1 <- causal.effect(y = "Z", x = "X", z = "W", G = gml1, expr = TRUE)
R> cat(cond1)

\frac{P(Z|W,X)}{\left(\sum_{Z}P(Z|W,X)\right)}
```

In mathematical notation the result reads

$$\frac{P(z|w,x)}{\sum_z[P(z|w,x)]}.$$

This is a typical case where the resulting expression is slightly awkward due to the incompleteness of the simplification rules. However, in this case it is easy to see that the expression can be simplified into $P(z|w,x)$.

5.2. A complex expression

The conditional distributions $P(v_i|v_\pi^{(i-1)})$ that are computed on line 6 can sometimes produce difficult expressions when causal effects are determined from complex graphs. This is a result

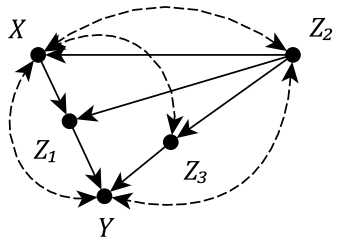


Figure 7: An example of a graph, where an identifiable causal effect results in a complex expression.

of the simplification rules which were described in the previous section, and their inability to handle every situation. The graph G of Figure 7 serves to demonstrate this phenomenon. An attempt is made to identify $P_x(z_1, z_2, z_3, y)$ in this graph.

Tian (2002) proved this effect to be identifiable, and showed that its expression is

$$P_x(z_1, z_2, z_3, y) = P(z_1|x, z_2) \sum_x P(y, z_3|x, z_1, z_2)P(x, z_2).$$

When applying Algorithm 1 to this causal effect, it is necessary to compute a conditional distribution $P^*(Y|Z_2, Z_3)$, where

$$P^*(y, z_2, z_3) = \sum_x P(y|z_2, x, z_3, z_1)P(z_3|z_2, x)P(x|z_2)P(z_2)$$

and P is the joint distribution of the observed variables of G. Now, the function `causal.effect` is applied as follows.

```
R> fig7 <- graph.formula(X -+ Z_1, Z_1 -+ Y, Z_3 -+ Y, Z_2 -+ X,
+    Z_2 -+ Z_1, Z_2 -+ Z_3, X -+ Y, Y -+ X, X -+ Z_3, Z_3 -+ X,
+    X -+ Z_2, Z_2 -+ X, Y -+ Z_2, Z_2 -+ Y, simplify = FALSE)
R> fig7 <- set.edge.attribute(graph = fig7, name = "description",
+    index = 7:14, value = "U")
R> ce3 <- causal.effect(y = c("Z_1", "Z_2", "Z_3", "Y"), x = "X",
+    z = NULL, G = fig7, expr = TRUE)
R> cat(ce3)
```

This results in the expression

$$P(z_1|z_2, x)\frac{(\sum_x P(y|z_2, x, z_3, z_1)P(z_3|z_2, x)P(x|z_2)P(z_2))}{\left(\sum_{x,y} P(y|z_2, x, z_3, z_1)P(z_3|z_2, x)P(x|z_2)P(z_2)\right)} \times \left(\sum_{x,z_3,y} P(y|z_2, x, z_3, z_1)P(z_3|z_2, x)P(x|z_2)P(z_2)\right) P(z_3|z_2)$$

This result is clearly more cumbersome than the one determined by Tian. However, it can be shown that this expression is correct by using do-calculus. Because the set $\{X, Z_2\}$ d-separates

all paths from Z_1 to Z_3, it follows that $(Z_3 \perp\!\!\!\perp Z_1 | X, Z_2)_G$, so

$$\begin{aligned} & P(z_1|z_2, x) \sum_x P(y|z_2, x, z_3, z_1) P(z_3|z_2, x) P(x|z_2) P(z_2) \\ = & P(z_1|z_2, x) \sum_x P(y|z_2, x, z_3, z_1) P(z_3|z_2, x, z_1) P(x, z_2) \\ = & P(z_1|z_2, x) \sum_x P(y, z_3|z_2, x, z_1) P(x, z_2), \end{aligned}$$

where the second equality is due to the conditional independence of Z_1 and Z_3 given X and Z_2. The last line is equivalent with Tian's expression up to the ordering of terms. It can be shown, that the remaining terms are subtracted from the expression.

$$\begin{aligned} & \frac{P(z_3|z_2) \sum_{x,z_3,y} P(y|z_2, x, z_3, z_1) P(z_3|z_2, x) P(x|z_2) P(z_2)}{\sum_{x,y} P(y|z_2, x, z_3, z_1) P(z_3|z_2, x) P(x|z_2) P(z_2)} \\ = & \frac{P(z_3|z_2) P(z_2)}{\sum_{x,y} P(y|z_2, x, z_3, z_1) P(z_3|z_2, x) P(x, z_2)}. \end{aligned}$$

By applying the same logic to the denominator, it follows that

$$\frac{P(z_3|z_2) P(z_2)}{\sum_{x,y} P(y|z_2, x, z_3, z_1) P(z_3|z_2, x) P(x, z_2)} = \frac{P(z_3|z_2) P(z_2)}{\sum_{x,y} P(y, z_3|z_2, x, z_1) P(x, z_2)}.$$

By using the conditional independence of Z_1 and Z_3 given X and Z_2 one gets

$$\begin{aligned} & \frac{P(z_3, z_2)}{\sum_x P(z_3|z_2, x, z_1) P(x, z_2)} = \frac{P(z_3, z_2)}{\sum_x P(z_3|z_2, x) P(x, z_2)} \\ = & \frac{P(z_3, z_2)}{\sum_x P(z_3|z_2, x) P(x, z_2)} = \frac{P(z_3, z_2)}{\sum_x P(z_3, z_2, x)} = \frac{P(z_3, z_2)}{P(z_3, z_2)} = 1. \end{aligned}$$

The expression produced by `causal.effect` is correct despite its complexity.

5.3. d-separation

Algorithm 1 does not utilize every possible independence property of a given graph G. For example, the conditional distribution of line six is conditioned on all nodes preceding V_i in the topological ordering π, even though at least some nodes on paths preceding V_i are often d-separated by some sets of nodes. In these cases, the nodes that are d-separated with V_i could be excluded from the expression, because they are conditionally independent from V_i in G. This situation is demonstrated by determining the expression of $P_{x,w}(y)$ in the graph G of Figure 8.

The function `causal.effect` is utilized

```
R> fig8 <- graph.formula(z -+ x, z -+ w, x -+ y, w -+ y)
R> ce3 <- causal.effect(y = "y", x = c("x", "w"), z = NULL, G = fig8,
+    expr = TRUE)
R> cat(ce3)
```

The function returns $P(y|x, w)$ even though Algorithm 1 would return $P(y|z, x, w)$. This is possible because $(Y \perp\!\!\!\perp Z | X, W)_G$. This means that our implementation is able to simplify the expression into $P(y|x, w)$.

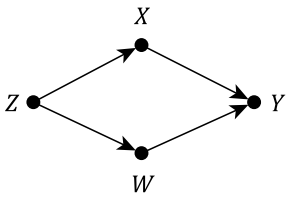


Figure 8: An example of a graph with additional conditional independences.

6. Discussion

We have introduced R package **causaleffect** for deriving expressions of joint interventional distributions in causal models. The task is a specific but important part of causal inference. We believe that our implementation has two practical use cases. First, **causaleffect** can be simply used to derive expressions of interventional distributions for complex causal models or to check manual derivations. This is an important step in the estimation of causal effects in complicated settings (Karvanen 2015). Second, **causaleffect** can be used as a building block in simulation studies and automated systems where identifiability needs to be checked for a large number of causal models. An example of this kind usage is already given by Hyttinen *et al.* (2015).

The efficiency of the presented implementation **causaleffect** could be analyzed further for example by simulation studies. However, an attempt to maximize performance was made by utilizing the most efficient packages available for the processing of graph files and for the objects corresponding to them. The existing simplification rules of the expressions could also be further improved, but it should be noted that sometimes the more complex expression can prove useful.

There have been many recent developments in the field of causality resulting in graph theoretic algorithms similar to **ID** and **IDC**. These include for example:

- Causal effect z-identifiability algorithm ID^{Z} (Bareinboim and Pearl 2012). z-identifiability deals with a situation, where it is possible to utilize a set $\mathbf{Z}$ that is disjoint from $\mathbf{X}$ to achieve identifiability.
- Causal effect transportability algorithm sID (Bareinboim and Pearl 2013a). Transportability means, that results obtained from experimental data can be generalized into a larger population, where only observational studies are applicable.
- Causal effect meta-transportability algorithm μsID (Bareinboim and Pearl 2013b). Meta-transportability is an extension of the concept of transportability, where the results are to be generalized from multiple experimental studies simultaneously.
- Counterfactual and conditional counterfactual identifiability algorithms ID* and IDC* (Shpitser and Pearl 2007).

The work presented in this paper could be utilized to implement these algorithms.

References

Bareinboim E, Pearl J (2012). "Causal Inference by Surrogate Experiments: z-Identifiability." In N de Freitas, K Murphy (eds.), *Proceedings of the Twenty-Eight Conference on Uncertainty in Artificial Intelligence*, pp. 113–120. AUAI Press.

Bareinboim E, Pearl J (2013a). "A General Algorithm for Deciding Transportability of Experimental Results." *Journal of Causal Inference*, **1**, 107–134. doi:10.1515/jci-2012-0004.

Bareinboim E, Pearl J (2013b). "Meta-Transportability of Causal Effects: A Formal Approach." In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 135–143.

Blackwell M (2015). ***causalsens**: Selection Bias Approach to Sensitivity Analysis for Causal Effects*. R package version 0.1.1, URL https://CRAN.R-project.org/package=causalsens.

Bontempi G, Olsen C, Flauder M (2015). ***D2C**: Predicting Causal Direction from Dependency Features*. R package version 1.2.1, URL https://CRAN.R-project.org/package=D2C.

Brandes U, Eiglsperger M, Herman I, Himsolt M, Marshall MS (2002). "GraphML Progress Report Structural Layer Proposal." In *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, pp. 501–512. Springer-Verlag. doi:10.1007/3-540-45848-4_59.

Carnegie NB, Harada M, Hill J (2016). ***treatSens**: A Package to Assess Sensitivity of Causal Analyses to Unmeasured Confounding*. New York. R package version 2.0.1, URL https://CRAN.R-project.org/package=treatSens.

Csardi G, Nepusz T (2006). "The **igraph** Software Package For Complex Network Research." *InterJournal*, **Complex Systems**, 1695. doi:10.1142/s0219525914500064.

Dandine-Roulland C (2015). ***ASPBay**: Bayesian Inference on Causal Genetic Variants Using Affected Sib-Pairs Data*. R package version 1.2, URL https://CRAN.R-project.org/package=ASPBay.

Dawid AP (1979). "Conditional Independence in Statistical Theory." *Journal of the Royal Statistical Society B*, **41**, 1–31.

Ding P (2012). ***ImpactIV**: Identifying Causal Effect for Multi-Component Intervention Using Instrumental Variable Method*. R package version 1.0, URL https://CRAN.R-project.org/package=ImpactIV.

Glynn A, Quinn K (2010). ***CausalGAM**: Estimation of Causal Effects with Generalized Additive Models*. R package version 0.1-3, URL https://CRAN.R-project.org/package=CausalGAM.

Huang Y, Valtorta M (2006). "Pearl's Calculus of Intervention Is Complete." In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pp. 217–224. AUAI Press.

Hyttinen A, Eberhardt F, Järvisalo M (2015). "Do-Calculus When the True Graph Is Unknown." In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, pp. 395–404. AUAI Press.

Kalisch M, Mächler M, Colombo D, Maathuis MH, Bühlmann P (2012). "Causal Inference Using Graphical Models with the R Package **pcalg**." *Journal of Statistical Software*, **47**(11), 1–26. doi:10.18637/jss.v47.i011.

Karvanen J (2015). "Study Design in Causal Models." *Scandinavian Journal of Statistics*, **42**(2), 361–377. doi:10.1111/sjos.12110.

Kashin K, Glynn A, Ichino N (2014). ***qualCI**: Causal Inference with Qualitative and Ordinal Information on Outcomes*. R package version 0.1, URL https://CRAN.R-project.org/package=qualCI.

Kim IS, Imai K (2014). ***wfe**: Weighted Linear Fixed Effects Regression Models for Causal Inference*. R package version 1.3, URL https://CRAN.R-project.org/package=wfe.

King G, Lucas C, Nielsen R (2015). ***MatchingFrontier**: R Package for Computing the Matching Frontier*. R package version 1.0.0, URL https://CRAN.R-project.org/package=MatchingFrontier.

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

Luo X, Small D, shan Li C, Rosenbaum P (2011). ***cin**: Causal Inference for Neuroscience*. R package version 0.1, URL https://CRAN.R-project.org/package=cin.

Maler E, Paoli J, Sperberg-McQueen CM, Yergeau F, Bray T (2004). "Extensible Markup Language (XML) 1.0 (Third Edition)." *Technical report*, W3C. URL http://www.w3.org/TR/2004/REC-xml-20040204.

Marchetti GM, Drton M, Sadeghi K (2015). ***ggm**: Functions for Graphical Markov Models*. R package version 2.3, URL https://CRAN.R-project.org/package=ggm.

Meinshausen N (2016). ***InvariantCausalPrediction**: Invariant Causal Prediction*. R package version 0.6-0, URL https://CRAN.R-project.org/package=InvariantCausalPrediction.

Millstein J (2016). ***cit**: Causal Inference Test*. R package version 2.1, URL https://CRAN.R-project.org/package=cit.

Neto EC, Yandell BS (2014). ***qtlnet**: Causal Inference of QTL Networks*. R package version 1.3.6, URL https://CRAN.R-project.org/package=qtlnet.

Pearl J (1995). "Causal Diagrams for Empirical Research." *Biometrika*, **82**, 669–688. doi:10.1093/biomet/82.4.669.

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

Peters J, Ernest J (2015). ***CAM**: Causal Additive Model (CAM)*. R package version 1.0, URL https://CRAN.R-project.org/package=CAM.

Quinn KM (2012). ***SimpleTable**: Bayesian Inference and Sensitivity Analysis for Causal Effects from 2×2 and $2 \times 2 \times K$ Tables in the Presence of Unmeasured Confounding*. R package version 0.1-2, URL https://CRAN.R-project.org/package=SimpleTable.

Ratkovic M (2015). ***SVMMatch**: Causal Effect Estimation and Diagnostics with Support Vector Machines*. R package version 1.1, URL https://CRAN.R-project.org/package=SVMMatch.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Saul B (2015). ***inferference**: Methods for Causal Inference with Interference*. R package version 0.4.62, URL https://CRAN.R-project.org/package=inferference.

Schutte S, Donnay K (2015). ***mwa**: Causal Inference in Spatiotemporal Event Data*. R package version 0.4.1, URL https://CRAN.R-project.org/package=mwa.

Shpitser I, Pearl J (2006a). "Identification of Conditional Interventional Distributions." In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence (UAI2006)*, pp. 437–444. AUAI Press.

Shpitser I, Pearl J (2006b). "Identification of Joint Interventional Distributions in Recursive Semi-Markovian Causal Models." In *Proceedings of the 21st National Conference on Artificial Intelligence – Volume 2*, pp. 1219–1226. AAAI Press.

Shpitser I, Pearl J (2007). "What Counterfactuals Can Be Tested." In *Proceedings of Twenty Third Conference on Uncertainty in Artificial Intelligence*, pp. 352–359. Vancouver.

Simpson JE (2002). ***XPath** and **XPointer**: Locating Content in XML Documents*. O'Reilly, Sebastopol.

Tan Z, Shu H (2013). ***iWeigReg**: Improved Methods for Causal Inference and Missing Data Problems*. R package version 1.0.

Temple Lang D (2016). ***XML**: Tools for Parsing and Generating XML within R and S-PLUS*. R package version 3.98-1.5, URL https://CRAN.R-project.org/package=XML.

Textor J, Hardt J, Knüppel S (2011). "**DAGitty**: A Graphical Tool for Analyzing Causal Diagrams." *Epidemiology*, **22**, 745.

Tian J (2002). *Studies in Causal Reasoning and Learning*. Phd thesis, Department of Computer Science, University of California, Los Angeles.

Tian J, Pearl J (2002). "A General Identification Condition For Causal Effects." In *Proceedings of the 18th National Conference on Artificial Intelligence*, pp. 567–573. AAAI Press.

Tian J, Pearl J (2003). "On the Identification of Causal Effects." *Technical report*, Department of Computer Science, University of California, Los Angeles. R-290-L.

Tingley D, Yamamoto T, Hirose K, Keele L, Imai K (2014). "**mediation**: R Package for Causal Mediation Analysis." *Journal of Statistical Software*, **59**(5), 1–38. doi:10.18637/jss.v059.i05.

Verma TS (1993). "Graphical Aspects of Causal Models." *Technical report*, Department of Computer Science, University of California, Los Angeles. R-191.

A. Graphs, causal models and causal effects

A.1. Graphs

The definitions that are presented here follow those of (Koller and Friedman 2009). *Graph* is an ordered pair $G = \langle \mathbf{V}, \mathbf{E} \rangle$, where $\mathbf{V}$ and $\mathbf{E}$ are sets such that

$$\mathbf{E} \subset \{\{X, Y\} \mid X \in \mathbf{V}, Y \in \mathbf{V}, X \neq Y\}.$$

The elements of $\mathbf{V}$ are the nodes of G, and the elements of $\mathbf{E}$ are the edges of G. A graph $F = \langle \mathbf{V}', \mathbf{E}' \rangle$ is a *subgraph* of G if $\mathbf{V}' \subset \mathbf{V}$ and $\mathbf{E}' \subset \mathbf{E}$. This is denoted as $F \subset G$. A graph G is *directed* if the set $\mathbf{E}$ consists of ordered pairs (X, Y). In a directed graph, node V_2 is a *child* of node V_1 if G contains an edge from V_1 to V_2, which means that $(V_1, V_2) \in \mathbf{E}$. Respectively V_2 is a *parent* of V_1 if $(V_2, V_1) \in \mathbf{E}$. The child-parent relationship is often denoted as $V_1 \rightarrow V_2$, where V_1 is a parent of V_2 and V_2 is a child of V_1. This can also be notated as $V_2 \leftarrow V_1$.

Let $n \geq 1$, $\mathbf{V} = \{V_1, \ldots, V_n\}$ and $V_i \neq V_j$ for all $i \neq j$. If $n > 1$, then the graph $H = \langle \mathbf{V}, \mathbf{E} \rangle$ is a *path* if

$$\mathbf{E} = \{\{V_1, V_2\}, \{V_2, V_3\}, \ldots, \{V_{n-1}, V_n\}\}$$

or if

$$\mathbf{E} = \{\{V_1, V_2\}, \{V_2, V_3\}, \ldots, \{V_{n-1}, V_n\}, \{V_n, V_1\}\}.$$

In the first case, H is a path from V_1 to V_n. In the second case H is a *cycle*. If $n = 1$, then $H = \langle \{V_1\}, \emptyset \rangle$ is also a path. A path H is a *directed path* if all of its edges are directed and point to the same direction, which means that either

$$\mathbf{E} = \{(V_1, V_2), (V_2, V_3), \ldots, (V_{n-1}, V_n)\}$$

or

$$\mathbf{E} = \{(V_1, V_2), (V_2, V_3), \ldots, (V_{n-1}, V_n), (V_n, V_1)\}.$$

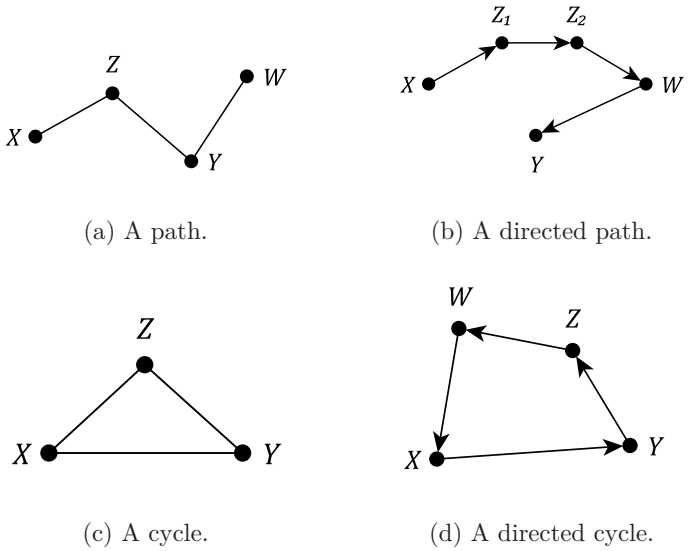
A node V_2 is a *descendant* of V_1 in G, if there exists a directed path H from V_1 to V_2 and $H \subset G$. Respectively, V_2 is an *ancestor* of V_1 in G, if there exists a directed path H from V_2 to V_1 and $H \subset G$. If a graph G does not contain any cycles, it is *acyclic*. A graph $G = \langle \mathbf{V}, \mathbf{E} \rangle$ is *connected* if there exists a path $H \subset G$ between every pair of nodes $V_i, V_j \in \mathbf{V}$. Examples of paths are cycles are presented in Figure 9.

If a graph is directed it is also possible to consider its subgraphs as undirected graphs, when all of the edges of the graph are regarded as undirected edges. For example, a directed graph contain paths, even if it does not contain any directed paths. The directed graph in Figure 10 contains a path connecting the nodes X and Y, even though they are not connected by a directed path.

Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be a graph and $\mathbf{Y} \subset \mathbf{V}$. Assume that the nodes of $\mathbf{Y}$ correspond to some observed variables, and that the set $\mathbf{V}$ can also contain nodes, which in turn correspond to some unobserved variables. Then the abbreviations $Pa(\mathbf{Y})_G$, $An(\mathbf{Y})_G$, and $De(\mathbf{Y})_G$ denote the set of observable parents, ancestors and descendants of the node set $\mathbf{Y}$ while also containing $\mathbf{Y}$.

A.2. Causal model

Causal model can be used to describe the functional relationships between variables of interest. In addition, the model enables the formal treatment of actions or interventions on the



(a) A path. (b) A directed path.

(c) A cycle. (d) A directed cycle.

Figure 9: Directed and undirected paths and cycles.

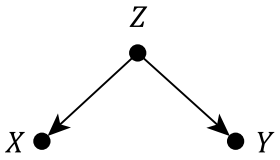


Figure 10: An undirected path in a directed graph.

variables of the model. Judea Pearl defined the deterministic causal model and its probabilistic counterpart (Pearl 2009, p. 203-205), which are presented in this section.

Definition 9 (Causal Model, (Pearl 2009) 7.1.1)**.** A *causal model* is a triple

$$M = \langle \mathbf{U}, \mathbf{V}, \mathbf{F} \rangle,$$

where:

1. $\mathbf{U}$ is a set of background variables that are determined by factors outside the model;
2. $\mathbf{V}$ is a set $\{V_1, V_2, \ldots, V_n\}$ of variables, called endogenous, that are determined by variables in the model – that is, variables in $\mathbf{U} \cup \mathbf{V}$; and
3. $\mathbf{F}$ is a set of functions $\{f_{V_1}, f_{V_2}, \ldots, f_{V_n}\}$ such that each f_{V_i} is a mapping from (the respective domains of) $\mathbf{U} \cup (\mathbf{V} \setminus V_i)$ to V_i, and such that the entire set $\mathbf{F}$ forms a mapping from $\mathbf{U}$ to $\mathbf{V}$. In other words, each f_i tells the value of V_i given the values of all other variables in $\mathbf{U} \cup \mathbf{V}$, and the entire set $\mathbf{F}$ has a unique solution $V(u)$. Symbolically, the set of equations $\mathbf{F}$ can be represented by writing

$$v_i = f_{V_i}(\mathbf{pa}_{V_i}, \mathbf{u}_{V_i}), \quad i = 1, \ldots, n,$$

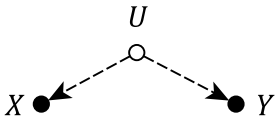


Figure 11: Example notation of unobserved edges.

Figure 12: Notation for bidirected edges.

where $\mathbf{pa}_i$ is any realization of the unique minimal set of variables $\mathbf{PA}_{V_i}$ in $\mathbf{V} \setminus V_i$ (connoting parents) sufficient for representing f_i. Likewise, $\mathbf{U}_{V_i} \subseteq \mathbf{U}$ stand for the unique minimal set of variables in $\mathbf{U}$ sufficient for representing f_i.

For each causal model M there is a corresponding graph $G = \langle \mathbf{W}, \mathbf{E} \rangle$. The node set $\mathbf{W}$ contains a node for each observed and unobserved variable of M. The edge set $\mathbf{E}$ is determined by the functional relationships between the variables of $\mathbf{V}$ and $\mathbf{U}$ in the causal model M. The set $\mathbf{E}$ contains an edge from X to Y if $X \in \mathbf{PA}_Y$, which means that there is an edge coming into V_i from every node required to uniquely define f_{V_i}. Likewise, the set $\mathbf{E}$ contains an edge from U to every node V_i such that $U \in \mathbf{U}_{V_i}$.

The definition of causal model does not set any limitations for the unobserved variables. Thus any unobserved node can be a parent of an arbitrary number of observed nodes. If every unobserved node is a parent of exactly two observed nodes, then the causal model is a *semi-Markovian causal model*. Verma (1993) showed, that for any causal model with unobserved variables one can construct a semi-Markovian causal model that encodes the same set of conditional independences. This is why only semi-Markovian models are considered in this paper.

The edges coming from unobserved variables are sometimes denoted as in Figure 11. However, it is common not to include the unobserved nodes in the visual representation of the graph, which serves to simplify the notation. Instead, it is said that there exists a *bidirected edge* between X and Y, which corresponds to the effect of the unobserved variable. Thus the notation of Figure 12 is utilized instead of the one in Figure 11.

This notation is used in (Huang and Valtorta 2006; Shpitser and Pearl 2006b; Tian 2002). It should be noted, that a bidirected edge is not the same as two directed edges between two nodes, as this would induce a cycle in the graph which is not allowed. Next, the definition of the causal model is expanded by defining a probability distribution for the unobserved variables.

Definition 10 (Probabilistic Causal Model, (Pearl 2009) 7.1.6)**.** A *Probabilistic causal model* is a pair

$$M = \langle M_D, P(\mathbf{U}) \rangle,$$

where M_D is a (deterministic) causal model and $P(\mathbf{U})$ is the joint distribution of the variables in $\mathbf{U}$.

Henceforth in the paper, the term causal model refers to a probabilistic semi-Markovian causal model without exception. Similarly, any graphs discussed will also refer to the graphs induced by these causal models. A graph G induced by a causal model is strongly related to the joint distribution P of all variables in the model, where $P = \prod_{i=1}^{n} P(v_i|pa^*(V_i)_G) \prod_{j=1}^{k} P(u_j)$, and $Pa^*(.)_G$ also contains all unobserved parents. If this relationship holds, then G is an *I-map* (independence map) of P. Independence properties of G and P are closely related through the following definition

Definition 11 (d-separation, (Pearl 2009) 1.2.3)**.** Let $H = \langle \mathbf{V}, \mathbf{E} \rangle$ be a path and a set $\mathbf{Z} \subset \mathbf{V}$. H is said to be *d-separated* by $\mathbf{Z}$ in G, if and only if either

1. H contains a chain $I \to M \to J$ or a fork $I \leftarrow M \to J$, where $M \in \mathbf{Z}$ and $I, J \in \mathbf{V}$., or
2. H contains an inverted fork $I \to M \leftarrow J$, where $De(M)_G \cap \mathbf{Z} = \emptyset$.

Disjoint sets $\mathbf{X}$ and $\mathbf{Y}$ are said to be d-separated by $\mathbf{Z}$ in G if every path from $\mathbf{X}$ to $\mathbf{Y}$ is d-separated by $\mathbf{Z}$ in G.

If $\mathbf{X}$ and $\mathbf{Y}$ are d-separated by $\mathbf{Z}$ in G, then $\mathbf{X}$ is independent of $\mathbf{Y}$ given $\mathbf{Z}$ in every P for which G is an I-map of P. The notation of (Dawid 1979) is used to denote this statement as $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})_G$.

A.3. Causal effects

Interventions on a causal model alter the functional relationships between its variables. Any intervention $do(\mathbf{X} = \mathbf{x})$ on a causal model M produces a new model $M_{\mathbf{x}} = \langle \mathbf{U}, \mathbf{V}, \mathbf{F}_{\mathbf{x}}, P(\mathbf{U}) \rangle$, where $\mathbf{F}_{\mathbf{x}}$ is obtained by replacing $f_X \in \mathbf{F}$ for each $X \in \mathbf{X}$ with a constant function, where the constants are defined as the $\mathbf{x}$ values of $do(\mathbf{X} = \mathbf{x})$. It is now feasible to formalize the notion of causal effects as follows.

Definition 12 (Causal Effect, (Shpitser and Pearl 2006b))**.** Let $M = \langle \mathbf{U}, \mathbf{V}, \mathbf{F}, P(\mathbf{U}) \rangle$ be a causal model and $\mathbf{Y}, \mathbf{X} \subset \mathbf{V}$. The *causal effect* of $do(\mathbf{X} = \mathbf{x})$ on the set $\mathbf{Y}$ in M is the marginal distribution of $\mathbf{Y}$ in $M_{\mathbf{x}}$, which is noted by $P(\mathbf{Y}|do(\mathbf{X} = \mathbf{x})) = P_{\mathbf{x}}(\mathbf{Y})$.

For every action $do(\mathbf{X} = \mathbf{x})$ it is required that $P(\mathbf{x}|Pa(\mathbf{X})_G \setminus \mathbf{X}) > 0$. This limitation ensures that $P_{\mathbf{x}}(\mathbf{V})$ and its marginals are well defined. The restriction stems from the fact that it is unfeasible to force $\mathbf{X}$ to attain values which cannot be observed. No inference can be made from the distribution of such an intervention using observational data.

Definition 13 (Causal Effect Identifiability, (Shpitser and Pearl 2006b) 2)**.** Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be a graph and $\mathbf{Y}, \mathbf{X} \subset \mathbf{V}$. The causal effect of $do(\mathbf{X} = \mathbf{x})$ on the set $\mathbf{Y}$, where $\mathbf{Y} \cap \mathbf{X} = \emptyset$, is *identifiable* in G if $P^1_{\mathbf{x}}(\mathbf{Y}) = P^2_{\mathbf{x}}(\mathbf{Y})$ for every pair of causal models M^1 and M^2 such that $P^1(\mathbf{V}) = P^2(\mathbf{V})$ and $P^1(\mathbf{x}|Pa(\mathbf{X})_G \setminus \mathbf{X}) > 0$.

It is often impossible to show that a causal effect is identifiable by using solely the definition, because one would have to compare every causal model that agree on the distribution of the

observed variables. However, the definition serves as a tool to prove unidentifiability in certain cases by constructing two causal models with the same induced graph and observational distribution, and by showing further that the interventional distributions differ. The reader is referred to (Shpitser and Pearl 2006b) for examples.

Affiliation:

Santtu Tikka
Department of Mathematics and Statistics
Faculty of Mathematics and Science
University of Jyvaskyla
P.O.Box 35, FI-40014, Finland
E-mail: santtu.tikka@jyu.fi

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
February 2017, Volume 76, Issue 12 *Submitted:* 2015-05-06
doi:10.18637/jss.v076.i12 *Accepted:* 2015-11-25

II

Simplifying probabilistic expressions in causal inference

Tikka, S. and Karvanen, J.

Journal of Machine Learning Research, 18(36): 1–30, 2017.

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

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, 2017). Another implementation of an identifiability algorithm can be found in the CIBN software by Jin Tian and 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

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

It turns out that there exists a significantly simpler expression,

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

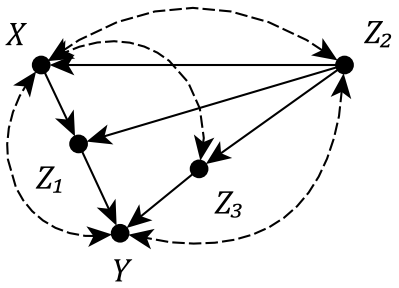


Figure 1: A graph for the introductory example on simplification.

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.

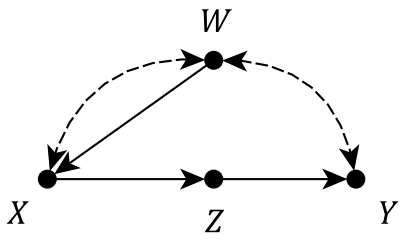


Figure 2: A graph G for the example on the use of simplification sets.

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

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

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

```
1:  function SIMPLIFY(A, G, π)
2:      j ← 0
3:      while j < |S| do
4:          B ← A
5:          J ← ∅
6:          D ← ∅
7:          R ← ∅
8:          I ← ∅
9:          j ← j + 1
10:         i ← INDEX.OF(A, j)
11:         M ← GET.MISSING(A, G, j)
12:         k ← 1
13:         while k ≤ i do
14:             ⟨J_new, D_new, R_new⟩ ← JOIN(J, D, V_k, C_k, S_j, M, G, π)
15:             if J_new ⊆ J then
16:                 break
17:             else
18:                 J ← J_new
19:                 D ← D_new
20:                 if R_new ≠ ∅ then
21:                     R ← R ∪ R_new
22:                     I ← I ∪ {D}
23:                     M ← M \ R_new
24:                 else
25:                     k ← k + 1
26:         if k = i + 1 then
27:             A_new ← FACTORIZE(J, D, R, I, A)
28:             if A_new = A then
29:                 A ← B
30:             else
31:                 A ← A_new
32:                 S ← S \ {S_j}
33:                 j ← 0
34:     return A
```

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

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** $\langle \mathbf{J}, \mathbf{D}, \emptyset \rangle$

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 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

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.

1: **function** INSERT($\mathbf{J}, \mathbf{D}, M', S, G, \pi$)
2: $\quad\mathbf{G} \leftarrow \mathbf{J}^{\pi} \setminus An^*(M')_G$
3: $\quad n \leftarrow |\mathbf{G}|$
4: $\quad$**for** $i = 1 : n$ **do**
5: $\quad\quad\mathbf{A} \leftarrow (An^*(M')_G \cup \mathbf{P}_i) \triangle \mathbf{D}$
6: $\quad\quad\mathbf{B} \leftarrow (An(M')_G \cup \mathbf{P}_i)$
7: $\quad\quad$**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**
8: $\quad\quad\quad$**return** $\langle \mathbf{J} \cup \{M'\}, (An^*(M')_G \cup \mathbf{P}_i), \{M'\}\rangle$
9: $\quad$**return** $\langle \mathbf{J}, \mathbf{D}, \emptyset\rangle$

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

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

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, π)
2:      B ← DECONSTRUCT(B, G, π)
3:      if S = ∅ then
4:          for X ∈ B do
5:              X ← EXTRACT(X, G, π)
6:          for ⟨T_A, S_A⟩ ∈ A do
7:              if S_A ≠ ∅ then
8:                  A_E ← ∅
9:                  R ← ∅
10:                 for ⟨V, C⟩ ∈ T_A do
11:                     if V ∉ S_A and C ∩ S_A = ∅ then
12:                         A_E ← A_E ∪ {⟨{⟨V, C⟩}, ∅⟩}
13:                         R ← R ∪ {⟨V, C⟩}
14:                 A ← A ∪ A_E
15:                 T_A ← T_A \ R
16:     else
17:         A_E ← ∅
18:         R ← ∅
19:         for ⟨T_A, S_A⟩ ∈ A do
20:             if S_A = ∅ then
21:                 T_A^(1) ← ∅
22:                 T_A^(2) ← ∅
23:                 for ⟨V, C⟩ ∈ T_A do
24:                     T_A^(1) ← T_A^(1) ∪ {V}
25:                     T_A^(2) ← T_A^(2) ∪ C
26:                 if T_A^(1) ∩ S = ∅ and T_A^(2) ∩ S = ∅ then
27:                     A_E ← A_E ∪ {⟨T_A, S_A⟩}
28:                     R ← R ∪ {⟨T_A, S_A⟩}
29:         A ← A \ R
30:         B_E ← {B}
31:         return ⟨B_E, A_E, ∅⟩
```

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 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.

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.

function q-SIMPLIFY(B_1, B_2, G, π)
  $B_1 \leftarrow$ EXTRACT(B_1, G, π)
  $B_2 \leftarrow$ EXTRACT(B_2, G, π)
  if $\mathbf{S}_1 \neq \emptyset$ **or** $\mathbf{S}_2 \neq \emptyset$ **then**
    return $\langle B_1, B_2 \rangle$
  $i \leftarrow 1$
  while $i \leq |\mathbf{B}_1|$ **and** $|\mathbf{B}_1| > 0$ **and** $|\mathbf{B}_2| > 0$ **do**
    for $j = 1 : |\mathbf{B}_2|$ **do**
      if $B_{1i} = B_{2j}$ **then**
        $\mathbf{B}_1 \leftarrow \mathbf{B}_1 \setminus \{B_{1i}\}$
        $\mathbf{B}_2 \leftarrow \mathbf{B}_2 \setminus \{B_{2j}\}$
        $i \leftarrow 0$
        break
    $i \leftarrow i + 1$
  $i \leftarrow 1$
  while $i \leq |\mathbf{A}_1|$ **and** $|\mathbf{A}_1| > 0$ **and** $|\mathbf{A}_2| > 0$ **do**
    for $j = 1 : |\mathbf{A}_2|$ **do**
      if $A_{1i} = A_{2j}$ **then**
        $\mathbf{A}_1 \leftarrow \mathbf{A}_1 \setminus \{A_{1i}\}$
        $\mathbf{A}_2 \leftarrow \mathbf{A}_2 \setminus \{A_{2j}\}$
        $i \leftarrow 0$
        break
    $i \leftarrow i + 1$
  return $\langle B_1, B_2 \rangle$

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.

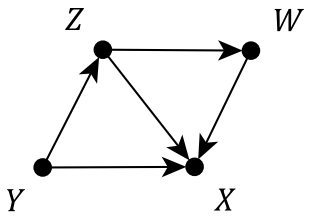


Figure 3: A graph G for the example on the necessity of the insertion procedure.

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

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

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 SIMPLIFY(A, G, π) will first attempt simplification in terms of Z, by calling

$$\text{JOIN}(\emptyset, \emptyset, X, \{W, Z, Y\}, Z, \{W\}, G, \pi),$$

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

$$\text{JOIN}(\{X\}, \{W, Z, Y\}, Z, Y, Z, \{W\}, G, \pi)$$

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

$$\text{INSERT}(\{X\}, \{W, Z, Y\}, W, Z, G, \pi).$$

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

$$\text{JOIN}(\emptyset, \emptyset, X, \{W, Z, Y\}, Y, \{W\}, G, \pi),$$

$$\text{JOIN}(\{X\}, \{W, Z, Y\}, Z, \{Y\}, Y, \{W\}, G, \pi),$$

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

$$\text{INSERT}(\{X\}, \{W, Z, Y\}, W, Y, G, \pi).$$

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, 2017). 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 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

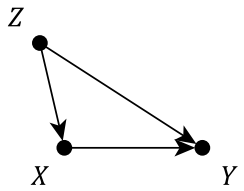


Figure 4: A graph G for a situation where simplification fails

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 **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 **FAIL**(F, F').

function IDC$(\mathbf{y}, \mathbf{x}, \mathbf{z}, P, G)$
1: **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$(\mathbf{y}, \mathbf{x} \cup \{z\}, \mathbf{z} \setminus \{z\}, P, G)$.
2: **else let** $P' =$ **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 $\mathrm{ID}(\{Y, Z_1, Z_2, Z_3\}, \{X\}, P(\mathbf{V}), G)$ 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 $\mathrm{ID}(\{Z_1\}, \{Y, Z_3, X, Z_2\}, P(\mathbf{V}), G)$ fires line 2 because Z_3 and Y are not ancestors of Z_1. The next call $\mathrm{ID}(\{Z_1\}, \{X, Z_2\}, P(Z_1, X, Z_2), G[\{Z_1, X, Z_2\}])$ 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 $\mathrm{ID}(\{Z_3\},\{Y,Z_1,X,Z_2\},P(\mathbf{V}),G)$ which also fires line 2 because X, Z_1 and Y and not ancestors of Z_3. Calling $\mathrm{ID}(\{Z_3\},\{Z_2\},P(Z_3,Z_2),G[\{Z_3,Z_2\}])$ 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 $\mathrm{ID}(\{Y,Z_2\},\{Z_3,Z_1,X\},P(\mathbf{V}),G)$. 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

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

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

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

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

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

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

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

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:

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

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, 2017.

Wolfram Research Inc. Mathematica, version 10.3, 2015.

III

Enhancing identification of causal effects by pruning

Tikka, S. and Karvanen, J.

Journal of Machine Learning Research, 18(194): 1–23, 2018.

Journal of Machine Learning Research 19 (2018) 1-23 Submitted 9/17; Revised 5/18; Published 5/18

Enhancing Identification of Causal Effects by Pruning

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

Causal models communicate our assumptions about causes and effects in real-world phenomena. Often the interest lies in the identification of the effect of an action which means deriving an expression from the observed probability distribution for the interventional distribution resulting from the action. In many cases an identifiability algorithm may return a complicated expression that contains variables that are in fact unnecessary. In practice this can lead to additional computational burden and increased bias or inefficiency of estimates when dealing with measurement error or missing data. We present graphical criteria to detect variables which are redundant in identifying causal effects. We also provide an improved version of a well-known identifiability algorithm that implements these criteria.

Keywords: causal inference, identifiability, causal model, pruning, algorithm

1. Introduction

A formal framework for causal inference is provided by the probabilistic causal model (Pearl, 2009) that encodes our knowledge of the variables of interest and their mutual relationships. In observational studies experimentation is not available, but through the causal model framework we can still symbolically intervene on variables, forcing them to take certain values as if an experiment had taken place. The question is whether we can make inferences about the effect of the intervention in the post-intervention model using only the observed probability distribution of the variables in the model before the intervention took place. This question is formally defined as identifiability of causal effects, and it has received considerable attention in literature, including a number of algorithmic solutions (Huang and Valtorta, 2006; Shpitser and Pearl, 2006; Tian and Pearl, 2002).

A causal model can be associated with a directed acyclic graph (DAG) that represents the functional relationships of the variables included in the model. The graphical representation provides us with the concept of d-separation (Geiger et al., 1990), that can be used to infer conditional independences between variables from the graph. If the distribution of the variables implies no conditional independence statements other than those already encoded in the graph, we say that the distribution is faithful (Spirtes et al., 2000).

The use of d-separation in the post-intervention model is the basis of do-calculus (Pearl, 1995), which consists of a set of inference rules for manipulating interventional distributions. The purpose of do-calculus is to derive formulas for causal effects and other causal queries, and it has been shown to be complete with respect to the identifiability of causal effects

(Huang and Valtorta, 2006; Shpitser and Pearl, 2006). The derived formulas provide recipes for estimating the causal effects from observational data.

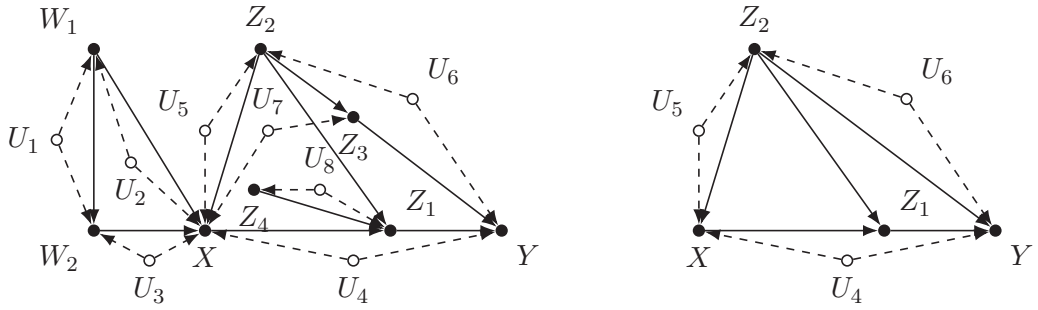
When computing causal effect formulas, we often apply an identifiability algorithm, such as the ID algorithm by Shpitser and Pearl (2006). Criteria for identifiability such as the back-door criterion and front-door criterion are available for manual derivations (Pearl, 2009) but the ID algorithm is more general and thus more suitable for automated processing. The ID algorithm splits the original problem into smaller subproblems which are then solved and aggregated as the final expression for the causal effect.

Complicated expressions are likely to arise in situations where we have included variables in our model that do not provide further benefit for the identification of the causal effect of interest. It is often the case that these variables nonetheless appear in the resulting formula, and deriving a simpler expression with the variable eliminated can be non-trivial. It is hard to specify what makes one expression simpler than another, but we can consider a number of criteria to evaluate simplicity. For example, we can compare the number of sums and fractions and the number of variables present in the expression.

In this paper we propose a number of graphical criteria to infer which variables in our causal model are in fact not necessary for identification. These criteria allow us to prune the graph, which in practice means removing specific vertices and considering identification in a latent projection. A significantly simpler expression can be obtained by pruning alone, but we may also combine pruning with simplification procedures that operate symbolically on the interventional distribution as presented in (Tikka and Karvanen, 2017b). Applying these methods in conjunction often provides additional benefits.

We present an identifiability algorithm that is able to recognize and eliminate unnecessary variables from the graph based on our criteria resulting in a simpler expression. When a large number of graphs and identifiability queries are processed, evaluating simpler expressions has apparent computational benefits. First, it is more efficient to evaluate a simpler expression repeatedly especially when some variables have been completely removed which further reduces the complexity of the task. Second, in practical applications that involve real-world data, variables often contain missing data or are affected by bias. Obtaining expression that do not involve such variables can be of great benefit in estimation. Third, a simpler expression is easier to communicate.

An introductory example motivates the use of the improved algorithm. We are interested in the causal effect of X on Y in graph G of Figure 1(a). Here, open circles denote unobserved variables. A more in-depth overview of graph theoretic concepts used in this paper is provided in Section 2. The causal effect is identifiable and the output of the ID algorithm is



(a) Graph G. (b) Graph G after pruning.

Figure 1: Graph G before and after pruning for the introductory example.

$$\sum_{z_2,z_4,z_3,z_1} \left(\sum_{w_1,w_2,x'} P(y|w_1,z_2,z_4,w_2,z_3,x',z_1)P(x'|w_1,z_2,z_4,w_2,z_3) \times P(z_3|w_1,z_2,z_4,w_2)P(w_2|w_1,z_2,z_4)P(z_2|w_1)P(w_1) \right) \Big/ \left(\sum_{w_1,w_2,x',y'} P(y'|w_1,z_2,z_4,w_2,z_3,x',z_1)P(x'|w_1,z_2,z_4,w_2,z_3) \times P(z_3|w_1,z_2,z_4,w_2)P(w_2|w_1,z_2,z_4)P(z_2|w_1)P(w_1) \right) \times \left(\sum_{w_1,w_2,z_3',x',y'} P(y'|w_1,z_2,z_4,w_2,z_3',x',z_1)P(x'|w_1,z_2,z_4,w_2,z_3') \times P(z_3'|w_1,z_2,z_4,w_2)P(w_2|w_1,z_2,z_4)P(z_2|w_1)P(w_1) \right) \times P(z_1|w_1,z_2,z_4,w_2,x)P(z_3|z_2)P(z_4).$$

This expression is very cumbersome and complicated. However, it turns out that a simpler expression exists for the causal effect. By exploiting the structure of the graph and using standard probability calculus the following expression can be obtained

$$\sum_{z_2,z_1} \left(\sum_{x'} P(y|z_2,z_1,x')P(x'|z_2)P(z_2) \right) P(z_1|z_2,x).$$

This expression is simpler in every regard compared to the original output. It contains fewer terms and no fractions. Also, we have completely removed the variables w_1, w_2 and z_4 from the expression. It can be shown that identifying the causal effect in the original graph is

equivalent to identifying it in the graph depicted in Figure 1(b). By running our improved algorithm we are able to prune the original graph and obtain this simpler expression directly. The algorithm works recursively and the pruning is carried out at each stage of the recursion. The recursive pruning provides significant benefits over pruning as a pre-processing step as demonstrated later.

The paper is structured as follows. In Section 2 we review crucial definitions and concepts related to graph theory and causal models. In Section 3 we focus on semi-Markovian causal models and present the original formulation of the ID algorithm. Our main results are presented in Section 4 and they are implemented into an improved identifiability algorithm in Section 5. Examples on the benefits of recursive pruning are provided in Section 6. Section 7 concludes with a discussion.

2. Definitions

We assume the reader to be familiar with a number of graph theoretic concepts and refer them to works such as (Koller and Friedman, 2009). We use capital letters to denote vertices and the respective variables, and small letters to denote their values. Bold letters are used to denote sets. A directed graph with a vertex set $\mathbf{V}$ and an edge set $\mathbf{E}$ is denoted by $\langle \mathbf{V}, \mathbf{E} \rangle$. For a graph $G = \langle \mathbf{V}, \mathbf{E} \rangle$ and a set of vertices $\mathbf{W} \subseteq \mathbf{V}$ the sets $\mathrm{Pa}(\mathbf{W})_G, \mathrm{Ch}(\mathbf{W})_G, \mathrm{An}(\mathbf{W})_G$ and $\mathrm{De}(\mathbf{W})_G$ denote a set that contains $\mathbf{W}$ in addition to its parents, children, ancestors and descendants in G, respectively. We also define the set $\mathrm{Co}(\mathbf{W})_G$ to denote the set of vertices that are connected to $\mathbf{W}$ in G via paths where the directionality of the edges is ignored, including $\mathbf{W}$. The root set of a graph G is the set of vertices without any descendants $\{X \in V \mid \mathrm{De}(X)_G \setminus \{X\} = \emptyset\}$, where $\setminus$ denotes the set difference. A subgraph of a graph $G = \langle \mathbf{V}, \mathbf{E} \rangle$ induced by a set of vertices $\mathbf{W} \subset \mathbf{V}$ is denoted by $G[\mathbf{W}]$. This subgraph retains all edges $V \to W$ of G such that $V, W \in \mathbf{W}$. The graph obtained from G by removing all incoming edges of $\mathbf{X}$ and all outgoing edges of $\mathbf{Z}$ is written as $G_{\overline{\mathbf{X}},\underline{\mathbf{Z}}}$. To facilitate analysis of causal effects we must first define the probabilistic causal model (Pearl, 2009).

Definition 1 (Probabilistic Causal Model) *A* probabilistic causal model *is a quadruple*

$$M = \langle \mathbf{U}, \mathbf{V}, \mathbf{F}, P(\mathbf{u}) \rangle,$$

where

1. $\mathbf{U}$ *is a set of unobserved (exogenous) variables that are determined by factors outside the model.*

2. $\mathbf{V}$ *is a set* $\{V_1, V_2, \ldots, V_n\}$ *of observed (endogenous) variables that are determined by variables in* $\mathbf{U} \cup \mathbf{V}$.

3. $\mathbf{F}$ *is a set of functions* $\{f_{V_1}, f_{V_2}, \ldots, f_{V_n}\}$ *such that each* f_{V_i} *is a mapping from (the respective domains of)* $\mathbf{U} \cup (\mathbf{V} \setminus \{V_i\})$ *to* V_i*, and such that the entire set* $\mathbf{F}$ *forms a mapping from* $\mathbf{U}$ *to* $\mathbf{V}$.

4. $P(\mathbf{u})$ *is a joint probability distribution of the variables in the set* $\mathbf{U}$.

Each causal model induces a causal diagram which is a directed graph that provides a graphical means to convey our assumptions of the causal mechanisms involved. The induced graph is constructed by adding a vertex for each variable in $\mathbf{U} \cup \mathbf{V}$ and a directed edge from $V_i \in \mathbf{U} \cup \mathbf{V}$ into $V_j \in \mathbf{V}$ whenever f_{V_j} is defined in terms of V_i.

Causal inference often focuses on a sub-class of models that satisfy additional assumptions: each $U \in \mathbf{U}$ appears in at most two functions of $\mathbf{F}$, the variables in $\mathbf{U}$ are mutually independent and the induced graph of the model is acyclic. Models that satisfy these additional assumptions are called *semi-Markovian causal models*. A graph associated with a semi-Markovian model is called a *semi-Markovian graph* (SMG). In SMGs every $U \in \mathbf{U}$ has at most two children. When semi-Markovian models are considered it is common not to depict background variables in the induced graph explicitly. Unobserved variables with exactly two children are not denoted as $V_i \leftarrow U \rightarrow V_j$ but as a bidirected edge $V_i \leftrightarrow V_j$ instead. Furthermore, unobserved variables with only one or no children are omitted entirely. We also adopt these abbreviations. For SMGs the sets $\mathrm{Pa}(\cdot)_G, \mathrm{Ch}(\cdot)_G, \mathrm{An}(\cdot)_G, \mathrm{De}(\cdot)_G$ and $\mathrm{Co}(\cdot)_G$ contain only observed vertices. Additionally, a subgraph $G[\mathbf{W}]$ of an SMG G will also retain any bidirected edges between vertices in $\mathbf{W}$.

Any DAG can be associated with an SMG by constructing its *latent projection* (Verma, 1993).

Definition 2 (latent projection) *Let $G = \langle \mathbf{V} \cup \mathbf{L}, \mathbf{E} \rangle$ be a DAG such that the vertices in $\mathbf{V}$ are observed and the vertices in $\mathbf{L}$ are latent. The* latent projection *$L(G, \mathbf{V})$ is a DAG $\langle \mathbf{V}, \mathbf{E}_L \rangle$, where for every pair of distinct vertices $Z, W \in \mathbf{V}$ it holds that:*

1. *$L(G, \mathbf{V})$ contains an edge $Z \rightarrow W$ if there exists a directed path $Z \rightarrow \cdots \rightarrow W$ in G on which every vertex except Z and W is in $\mathbf{L}$.*
2. *$L(G, \mathbf{V})$ contains an edge $Z \leftrightarrow W$ if there exists a path from Z to W in G that does not contain the pattern $Z \rightarrow M \leftarrow W$ (a collider) and on which every vertex except Z and W is in $\mathbf{L}$ and the first edge has an arrowhead pointing into W and the last edge has an arrowhead pointing into Z.*

From the construction it is easy to see that a latent projection is in fact an SMG. The induced graph of a probabilistic causal model can also be used to derive conditional independences among the variables in the model using a concept known as d-separation. We provide a definition for d-separation (Shpitser and Pearl, 2008) which takes into account the presence of bidirected edges and is thus suitable for SMGs.

Definition 3 (d-separation) *A path P in an SMG G is said to be d-separated by a set $\mathbf{Z}$ if and only if either*

1. *P contains one of the following three patterns of edges: $I \rightarrow M \rightarrow J$, $I \leftrightarrow M \rightarrow J$ or $I \leftarrow M \rightarrow J$, such that $M \in \mathbf{Z}$, or*
2. *P contains one of the following three patterns of edges: $I \rightarrow M \leftarrow J$, $I \leftrightarrow M \leftarrow J$, $I \leftrightarrow M \leftrightarrow J$, such that $De(M)_G \cap \mathbf{Z} = \emptyset$.*

Disjoint sets $\mathbf{X}$ and $\mathbf{Y}$ are said to be d-separated by $\mathbf{Z}$ in G if every path from $\mathbf{X}$ to $\mathbf{Y}$ is d-separated by $\mathbf{Z}$ in G.

Whenever we can decompose the joint distribution of the observed variables $\mathbf{V}$ and the unobserved variables $\mathbf{U}$ as $P(\mathbf{v},\mathbf{u}) = \prod_{W\in\mathbf{V}\cup\mathbf{U}} P(w|\mathrm{Pa}^*(w)_G)$, where $\mathrm{Pa}^*(\cdot)$ also contains the unobserved parents but not the argument itself, we say that G is an I-map of $P(\mathbf{v},\mathbf{u})$ (Pearl, 2009). If sets $\mathbf{X}$ and $\mathbf{Y}$ are d-separated by $\mathbf{Z}$ in G, then $\mathbf{X}$ is independent of $\mathbf{Y}$ given $\mathbf{Z}$ in every P for which G is an I-map (Pearl, 1988). We use the notation of (Dawid, 1979) to denote this d-separation and conditional independence statement as $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z})_G$. It is clear that the graph induced by any semi-Markovian causal model is an I-map for the joint distribution $P(\mathbf{v},\mathbf{u})$ induced by the model.

Our interest lies in the effects of actions imposing changes to the model. An action that forces $\mathbf{X}$ to take a specific value $\mathbf{x}$ is called an *intervention* and it is denoted by $\mathrm{do}(\mathbf{x})$ (Pearl, 2009). An intervention $\mathrm{do}(\mathbf{x})$ on a model M creates a new sub-model, denoted by $M_{\mathbf{x}}$, where the functions in $\mathbf{F}$ that determine the value of $\mathbf{X}$ have been replaced with constant functions. The *interventional distribution* of a set of variables $\mathbf{Y}$ in the model $M_{\mathbf{x}}$ is denoted by $P_{\mathbf{x}}(\mathbf{y})$. This distribution is also known as the *causal effect* of $\mathbf{X}$ on $\mathbf{Y}$.

Multiple causal models can share the same graph, and thus the same sub-model resulting from an intervention. The question is, are our assumptions encoded in the causal model sufficient to uniquely specify an interventional distribution of interest. This notion is captured by the following definition (Shpitser and Pearl, 2006).

Definition 4 (identifiability) *Let $G = \langle \mathbf{V}, \mathbf{E}\rangle$ be an SMG and let $\mathbf{X}$ and $\mathbf{Y}$ be disjoint sets of variables such that $\mathbf{X},\mathbf{Y} \subset \mathbf{V}$. The causal effect of $\mathbf{X}$ on $\mathbf{Y}$ is said to be* identifiable *from P in G if $P_{\mathbf{x}}(\mathbf{y})$ is uniquely computable from $P(\mathbf{V})$ in any causal model that induces G.*

In order to show the identifiability of a given effect we have to express the interventional distribution in terms of observed probabilities only. The link between observed probabilities and interventional distributions is provided by three inference rules known as *do-calculus* (Pearl, 1995):

1. Insertion and deletion of observations:

$$P_{\mathbf{x}}(\mathbf{y}|\mathbf{z},\mathbf{w}) = P_{\mathbf{x}}(\mathbf{y}|\mathbf{w}), \text{ if } (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}}}}.$$

2. Exchanging actions and observations:

$$P_{\mathbf{x},\mathbf{z}}(\mathbf{y}|\mathbf{w}) = P_{\mathbf{x}}(\mathbf{y}|\mathbf{z},\mathbf{w}), \text{ if } (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}},\underline{\mathbf{Z}}}}.$$

3. Insertion and deletion of actions:

$$P_{\mathbf{x},\mathbf{z}}(\mathbf{y}|\mathbf{w}) = P_{\mathbf{x}}(\mathbf{y}|\mathbf{w}), \text{ if } (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}},\overline{Z(\mathbf{W})}}},$$

   where $Z(\mathbf{W}) = \mathbf{Z} \setminus \mathrm{An}(\mathbf{W})_{G_{\overline{\mathbf{X}}}}$.

Completeness of do-calculus was established independently by Huang and Valtorta (2006) and Shpitser and Pearl (2006). In this paper we focus on the solution provided by Shpitser and Pearl (2006). They constructed an identifiability algorithm called ID, which in essence applies the rules of do-calculus and breaks the problem into smaller sub-problems repeatedly.

3. ID Algorithm

In order to present the ID algorithm, we first need some additional definitions that are used to construct the graphical criterion for non-identifiability (Shpitser and Pearl, 2006).

Definition 5 (C-component) *Let G be an SMG and let $C \subseteq G$. If every pair of vertices in C is connected by a bidirected path, that is a path consisting entirely of bidirected edges, then C is a* C-component *(confounded component). Furthermore, C is a* maximal C-component *if C contains every vertex connected to C via bidirected paths in G and C is an induced subgraph of G.*

No restrictions are imposed on the directed edges of a C-component. The same is not true for the maximal C-components (also known as districts) of an SMG G, which are assumed to be induced subgraphs of G. This requirement guarantees the uniqueness of the maximal C-components.

Maximal C-components are an important tool for identifying causal effects. The set of maximal C-components of a semi-Markovian graph G is denoted by $C(G)$. A result in (Tian, 2002) states that if $C = \langle \mathbf{C}, \mathbf{E} \rangle$ is a maximal C-component and $C \subset G$ then the causal effect $P_{\mathbf{v}\setminus\mathbf{c}}(\mathbf{c})$ is identifiable from P in G. A distribution P of a semi-Markovian model also factorizes with respect to the maximal C-components of the induced graph G such that $P(\mathbf{v}) = \prod_{\langle \mathbf{C},\mathbf{E}\rangle \in C(G)} P_{\mathbf{v}\setminus\mathbf{c}}(\mathbf{c})$ (Shpitser and Pearl, 2006). It is precisely this factorization that the ID algorithm takes advantage of. A specific type of C-component is used to characterize problematic structures for identifiability.

Definition 6 (C-forest) *Let G be an SMG and let $\mathbf{Y}$ be the root set of G. If G is a C-component and all observed vertices have at most one child, then G is a* $\mathbf{Y}-$rooted C-forest.

The complete criterion for non-identifiability uses a structure formed by two C-forests:

Definition 7 (hedge) *Let $\mathbf{X}, \mathbf{Y} \subset \mathbf{V}$ be disjoint sets of variables and let G be an SMG. Let $F = \langle \mathbf{V}_F, \mathbf{E}_F \rangle$ and $F' = \langle \mathbf{V}_{F'}, \mathbf{E}_{F'} \rangle$ be $\mathbf{R}$-rooted C-forests in G such that $\mathbf{V}_F \cap \mathbf{X} \neq \emptyset$, $\mathbf{V}_{F'} \cap \mathbf{X} = \emptyset$, $F' \subseteq F$, and $\mathbf{R} \subseteq An(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$. Then F and F' form a hedge for $P_{\mathbf{x}}(\mathbf{y})$ in G.*

Intuitively hedges are a difficult concept. Whenever a hedge is present, there exists two causal models with the same probability distribution over $\mathbf{V}$ but their interventional distributions do not agree. Observational data can not be used to estimate causal effects in this scenario. We are now ready to present the ID algorithm.

Shpitser and Pearl (2006) showed that whenever Algorithm 1 returns an expression for a causal effect, it is correct. Additionally whenever line 5 is triggered there exists a hedge for the causal effect currently being identified. This result establishes the completeness of the algorithm and also the completeness of do-calculus, since the soundness of each line of the algorithm can be shown with do-calculus and standard probability calculus alone.

Algorithm 1 The causal effect of intervention $do(\mathbf{X} = \mathbf{x})$ on $\mathbf{Y}$ (ID).

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

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

function ID$(\mathbf{y}, \mathbf{x}, P, G)$

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

4. Pruning of Variables

In this section we present a number of results that deal with variables that are not necessary for identification either by removing them from the graph or by considering them latent. When the causal effect $P_{\mathbf{x}}(\mathbf{y})$ is considered in an SMG we can present an outline of the pruning process:

1. Removal of non-ancestors of $\mathbf{Y}$.
2. Removal of ancestors of $\mathbf{X}$ that are connected to $\mathbf{Y}$ only via $\mathbf{X}$ under certain conditions.
3. Removal of vertices connected to other vertices only through a single vertex.
4. Identification in a latent projection under certain conditions.

Steps 2–4 are new and they are based on the results of this section. Step 1 is derived from a useful result by Shpitser and Pearl (2006) which states that for a causal effect $P_{\mathbf{x}}(\mathbf{y})$ we can always ignore non-ancestors of $\mathbf{Y}$.

Lemma 8 *Let* $\mathbf{X}' = \mathbf{X} \cap An(\mathbf{Y})_G$. *Then* $P_{\mathbf{x}}(\mathbf{y})$ *obtained from* P *in* G *is equal to* $P'_{\mathbf{x}'}(\mathbf{y})$ *obtained from* $P' = P(An(\mathbf{Y})_G)$ *in* $G[An(\mathbf{Y})_G]$.

Lemma 8 is implemented on line 2 of Algorithm 1. Not all ancestors of $\mathbf{Y}$ are always necessary for identification. The next result states that we may sometimes remove ancestors of $\mathbf{X}$ that are connected to $\mathbf{Y}$ only through $\mathbf{X}$.

Theorem 9 *Let G be an SMG and let $\mathbf{Z} \subset \mathbf{V}$ be the set of all vertices such that $\mathbf{X}$ intercepts all paths from $\mathbf{Z}$ to $\mathbf{Y}$. Then the causal effect $P_{\mathbf{x}}(\mathbf{y})$ obtained from P in G is equal to $P'_{\mathbf{x}}(\mathbf{y})$ obtained from $P' = P(\mathbf{V} \setminus \mathbf{Z})$ in $G[\mathbf{V} \setminus \mathbf{Z}]$ if $\mathbf{Z}$ contains no members of $\mathbf{X}$ and if $G[\mathbf{V} \setminus \mathbf{Z}] = L(G, \mathbf{V} \setminus \mathbf{Z})$.*

Proof Let $G' = G[\mathbf{V} \setminus \mathbf{Z}]$ and assume that $G' = L(G, \mathbf{V} \setminus \mathbf{Z})$. Let $\mathbf{U_Z}$, $\mathbf{U_{V \setminus Z}}$ and $\mathbf{U_X}$ be sets of unobserved variables such that for all $U \in \mathbf{U_Z}$ it holds that $\mathrm{Ch}(U)_{G_{\overline{\mathbf{X}}}} \subseteq \mathbf{Z}$, for all $U \in \mathbf{U_{V \setminus Z}}$ it holds that $\mathrm{Ch}(U)_{G_{\overline{\mathbf{X}}}} \subseteq \mathbf{V} \setminus \mathbf{Z}$ and for all $U \in \mathbf{U_X}$ it holds that $\mathrm{Ch}(U)_G \in \mathbf{X}$. The sets $\mathbf{U_Z}$, $\mathbf{U_{V \setminus Z}}$ and $\mathbf{U_X}$ partition $\mathbf{U}$ because $\mathbf{X}$ intercepts all paths from $\mathbf{Z}$ to $\mathbf{Y}$. According to the third rule of do-calculus $P_{\mathbf{x}}(\mathbf{y}) = P_{\mathbf{x},\mathbf{z}}(\mathbf{y})$ because the condition $(\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X})_{G_{\overline{\mathbf{X}}}}$ holds as removing the edges incoming to $\mathbf{X}$ separates $\mathbf{X}$ from its ancestors. Applying the truncated factorization formula (Pearl, 2009) we have that

$$P_{\mathbf{x},\mathbf{z}}(\mathbf{y}) = \sum_{\mathbf{U}} \sum_{\mathbf{V} \setminus (\mathbf{Y} \cup \mathbf{X} \cup \mathbf{Z})} \prod_{\mathbf{V} \setminus (\mathbf{X} \cup \mathbf{Z})} P(v_i | \mathrm{Pa}(v_i)_G \setminus \{v_i\}) \prod_{\mathbf{U}} P(u_i).$$

Since variables in $\mathbf{U_Z}$ can only be parents of variables in $\mathbf{Z}$ or $\mathbf{X}$ in G, we can sum them out from the previous expression and obtain

$$P_{\mathbf{x},\mathbf{z}}(\mathbf{y}) = \sum_{\mathbf{U_{V \setminus Z}} \cup \mathbf{U_X}} \sum_{\mathbf{V} \setminus (\mathbf{Y} \cup \mathbf{X} \cup \mathbf{Z})} \prod_{\mathbf{V} \setminus (\mathbf{X} \cup \mathbf{Z})} P(v_i | \mathrm{Pa}(v_i)_G \setminus \{v_i\}) \prod_{\mathbf{U_{V \setminus Z}} \cup \mathbf{U_X}} P(u_i).$$

Similarly, variables in $\mathbf{U_X}$ can only be parents of variables in $\mathbf{X}$ in G, so we can also sum them out of the expression to obtain

$$P_{\mathbf{x},\mathbf{z}}(\mathbf{y}) = \sum_{\mathbf{U_{V \setminus Z}}} \sum_{\mathbf{V} \setminus (\mathbf{Y} \cup \mathbf{X} \cup \mathbf{Z})} \prod_{\mathbf{V} \setminus (\mathbf{X} \cup \mathbf{Z})} P(v_i | \mathrm{Pa}(v_i)_G \setminus \{v_i\}) \prod_{\mathbf{U_{V \setminus Z}}} P(u_i).$$

We let $\mathbf{V}' = \mathbf{V} \setminus \mathbf{Z}$. Verma (1993) showed that a graph and its latent projection have the same set of conditional independence relations among the observed variables. Because we have assumed that $G' = L(G, \mathbf{V} \setminus \mathbf{Z})$ every conditional independence between variables in $\mathbf{V}'$ and $\mathbf{U_{V'}}$ applies in both G and G'. We have that for all $V_i \in \mathbf{V}' \setminus \mathbf{X}$ it holds that $P(v_i | \mathrm{Pa}(v_i)_G \setminus \{v_i\}) = P'(v_i | \mathrm{Pa}(v_i)_{G'} \setminus \{v_i\})$ and for all $U_i \in \mathbf{U_{V'}}$ it holds that $P(u_i) = P'(u_i)$. Finally we obtain

$$P_{\mathbf{x},\mathbf{z}}(\mathbf{y}) = \sum_{\mathbf{U_{V'}}} \sum_{\mathbf{V}' \setminus (\mathbf{Y} \cup \mathbf{X})} \prod_{\mathbf{V}' \setminus \mathbf{X}} P'(v_i | \mathrm{Pa}(v_i)_{G'} \setminus \{v_i\}) \prod_{\mathbf{U_{V'}}} P'(u_i) = P'_{\mathbf{x}}(\mathbf{y}).$$

■

Theorem 9 can also be applied in a more general setting where a subset of $\mathbf{X}$ intercepts all paths from a set $\mathbf{Z}$ to $\mathbf{Y}$

Corollary 10 *Let G be an SMG and let $\mathbf{Z} \subset \mathbf{V}$ be the set of all vertices such that a set $\mathbf{W} \subseteq \mathbf{X}$ intercepts all paths from $\mathbf{Z}$ to $\mathbf{Y}$ and no member of $\mathbf{X} \setminus \mathbf{W}$ is a descendant of $\mathbf{W}$. Then the causal effect $P_{\mathbf{x}}(\mathbf{y})$ obtained from P in G is equal to $P'_{\mathbf{x} \setminus \mathbf{z}}(\mathbf{y})$ obtained from $P' = P(\mathbf{V} \setminus \mathbf{Z})$ in $G[\mathbf{V} \setminus \mathbf{Z}]$ if $\mathbf{Z}$ contains no members of $\mathbf{W}$ and if $G[\mathbf{V} \setminus \mathbf{Z}] = L(G, \mathbf{V} \setminus \mathbf{Z})$.*

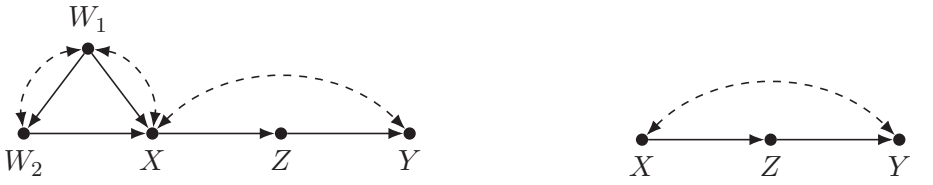
Proof Since $\mathbf{W}$ intercepts all paths from $\mathbf{Z}$ to $\mathbf{Y}$ and no member of $\mathbf{X} \setminus \mathbf{W}$ is a descendant of $\mathbf{W}$, it follows that no member of $\mathbf{W}$ is in $\mathbf{Z}$. According to the third rule of do-calculus we have that $(\mathbf{Y} \perp\!\!\!\perp \mathbf{Z} | \mathbf{X} \setminus \mathbf{Z})_{G_{\overline{\mathbf{X}\setminus\mathbf{Z}}}}$ and $P_{\mathbf{x}}(\mathbf{y}) = P_{\mathbf{x}\setminus\mathbf{z}}(\mathbf{y})$. The claim now follows by applying Theorem 9 to $P_{\mathbf{x}\setminus\mathbf{z}}(\mathbf{y})$. ■

Corollary 11 *When the causal effect $P_{\mathbf{x}}(\mathbf{y})$ is considered in graph G, a set of vertices $\mathbf{Z} = An(\mathbf{Y})_G \setminus Co(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$ can be removed from G if $G[\mathbf{V} \setminus \mathbf{Z}] = L(G, \mathbf{V} \setminus \mathbf{Z})$.*

Proof The set $\mathrm{An}(\mathbf{Y})_G$ contains $\mathbf{Y}$ in addition to the ancestors of $\mathbf{Y}$, and the set $\mathrm{Co}(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$ contains $\mathbf{Y}$ and all vertices that are connected to $\mathbf{Y}$ via a path that does not contain edges incoming to $\mathbf{X}$. Therefore, $\mathbf{Z}$ contains such ancestors of $\mathbf{Y}$ that all paths from $\mathbf{Z}$ to $\mathbf{Y}$ contain $\mathbf{X}$. The removal of $\mathbf{Z}$ from G is now licensed by Corollary 10. ■

Corollary 11 provides a constructive criterion for the set $\mathbf{Z}$ described in Corollary 10 when G consists only of $\mathbf{Y}$ and its ancestors. If a vertex Z_i is a member of $\mathrm{An}(\mathbf{Y})_G \setminus \mathrm{Co}(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$ then it must be connected to $\mathbf{Y}$ only through paths containing some $\mathbf{W}_{Z_i} \subseteq \mathbf{X}$. We can always choose the sets $\mathbf{W}_{Z_i}$ in such a way that the union $\mathbf{W} = \cup \mathbf{W}_{Z_i}$ over the members Z_i of $\mathrm{An}(\mathbf{Y})_G \setminus \mathrm{Co}(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$ has no descendants in $\mathbf{X} \setminus \mathbf{W}$. The set $\mathbf{W}$ intercepts all paths from $\mathbf{Z} = \cup\{Z_i\}$ to $\mathbf{Y}$. Conversely, if Z_i is a vertex such that a set $\mathbf{W} \subseteq \mathbf{X}$ intercepts all paths from Z_i to $\mathbf{Y}$, then Z_i cannot be connected to $\mathbf{Y}$ in $G_{\overline{\mathbf{X}}}$. If we assume that $G = G[\mathrm{An}(\mathbf{Y})_G]$ it follows that Z_i is a member of $\mathrm{An}(\mathbf{Y})_G \setminus \mathrm{Co}(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$.

We present a simple example to motivate the usefulness of Corollary 11. We apply the ID algorithm to identify the causal effect of X on Y in graph G of Figure 2(a). Applying



(a) Graph G. (b) Subgraph $G[\{X, Y, Z\}]$.

Figure 2: A graph for an example where Corollary 11 allows us to remove vertices W_1 and W_2 when the causal effect of X on Y is considered.

the ID algorithm results in the following expression for the causal effect

$$\sum_z P(z|w_1, w_2, x) \left(\sum_{w_1, w_2, x'} P(y|w_1, w_2, x', z) P(x'|w_1, w_2) P(w_2|w_1) P(w_1) \right).$$

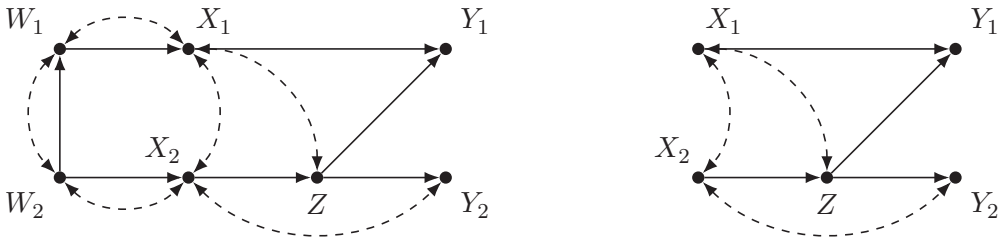
Applying Corollary 11 in this case would result in the removal of the vertices W_1 and W_2 from the graph, since they are ancestors of Y in G but not connected to Y in $G_{\overline{X}}$ and the corresponding latent projection is the subgraph $G[\{X, Y, Z\}]$ of Figure 2(b). Running the

ID algorithm in this subgraph provides us the following expression

$$\sum_z P(z|x) \left(\sum_{x'} P(y|x', z)P(x') \right).$$

We may consider this expression simpler compared to the previous output by noting that W_1 and W_2 do not appear in the expression and it has fewer unique terms. The same expression can also be obtained manually by applying the front-door criterion (Pearl, 2009).

Often the question of identifiability can not be answered directly by neither the back-door nor the front-door criterion which leads us to more general methods, such as the ID algorithm. We are interested in the causal effect of W_1, X_1 and X_2 on Y_1 and Y_2 in the graph of Figure 3(a).



(a) Graph G.

(b) Subgraph $G[\{X_1, X_2, Z, Y_1, Y_2\}]$.

Figure 3: A graph for an example where the back-door and front-door criteria are unavailable, but Corollary 11 allows us to remove vertices W_1 and W_2 even when W_1 is part of the intervention.

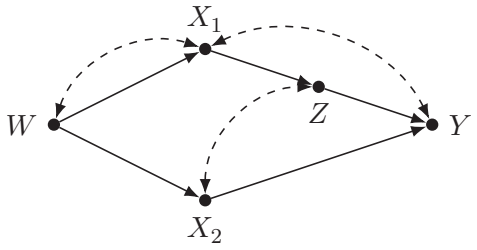
Direct application of the ID algorithm provides us with the following expression

$$\sum_z P(y_1|w_2, w_1, x_2, x_1, z)P(z|w_2, x_2) \left(\sum_{w_2, x_2'} P(y_2|w_2, x_2', z)P(x_2'|w_2)P(w_2) \right).$$

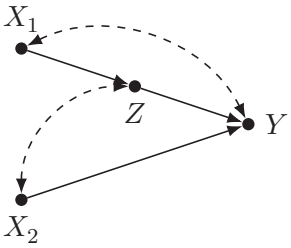
Corollary 11 licenses the removal of W_1 and W_2 from the graph. By running ID again in the resulting subgraph $G[\{X_1, X_2, Z, Y_1, Y_2\}]$ as shown in Figure 3(b) we obtain a simpler expression for the causal effect

$$\sum_z P(y_1|x_1, x_2, z)P(z|x_2) \left(\sum_{x_2'} P(y_2|x_2', z)P(x_2') \right).$$

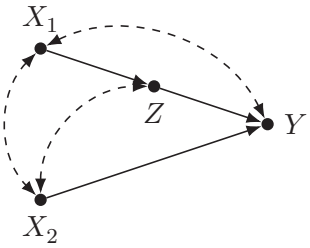
The next example illustrates the necessity of the assumption $G[\mathbf{V} \setminus \mathbf{Z}] = L(G, \mathbf{V} \setminus \mathbf{Z})$ of Theorem 9 and Corollary 10. We are interested in the causal effect of X_1 and X_2 on Y in graph G of Figure 4(a). In this graph W is connected to Y only through X_1 and X_2, but the corresponding latent projection does not match the subgraph with W removed as seen in Figures 4(b) and 4(c). In G the causal effect is identifiable, but it is not identifiable in the latent projection $G' = L(G, \{X_1, X_2, Z, Y\})$. In this latent projection a bidirected edge exists between X_1 and X_2 and a hedge is formed by the C-forests G' and $G[\{Y\}]$.



(a) Graph G.



(b) Subgraph $G[\{X_1, X_2, Z, Y\}]$.



(c) Latent projection $L(G, \{X_1, X_2, Z, Y\})$.

Figure 4: An example where the assumptions of Corollary 10 are not met because the subgraph in panel (b) and the latent projection in panel (c) differ from each other.

We may also remove sets of vertices that are connected to the rest of the graph only through a single vertex even when no intervention on the corresponding variable has taken place.

Theorem 12 *Let G be an SMG such that $G = G[An(\mathbf{Y})_G]$ for a set of vertices $\mathbf{Y}$ and let W be a vertex of G. If there exists a set $\mathbf{Z}$ such that $\mathbf{Z} \cap (\mathbf{Y} \cup \mathbf{X}) = \emptyset$ and $\mathbf{Z}$ is connected to $\mathbf{V} \setminus \mathbf{Z}$ only through W. Then the causal effect $P_{\mathbf{x}}(\mathbf{y})$ obtained from P in G is equal to $P'_{\mathbf{x}}(\mathbf{y})$ obtained from $P' = P(\mathbf{V} \setminus \mathbf{Z})$ in $G[\mathbf{V} \setminus \mathbf{Z}]$.*

Proof Let $G' = G[\mathbf{V} \setminus \mathbf{Z}]$ and let $\mathbf{U_Z}$ and $\mathbf{U_{V \setminus Z}}$ be sets of unobserved variables such that for all $U \in \mathbf{U_Z}$ it holds that $\mathrm{Ch}(U)_G \in \mathbf{Z} \cup \{W\}$ and for all $U \in \mathbf{U_{V \setminus Z}}$ it holds that $\mathrm{Ch}(U)_G \in (\mathbf{V} \setminus \mathbf{Z}) \cup \{W\}$. Sets $\mathbf{U_Z}$ and $\mathbf{U_{V \setminus Z}}$ partition $\mathbf{U}$ because $\mathbf{Z}$ is connected to $\mathbf{V} \setminus \mathbf{Z}$ only through W. Applying the truncated factorization formula yields

$$P_{\mathbf{x}}(\mathbf{y}) = \sum_{\mathbf{U}} \sum_{\mathbf{V} \setminus (\mathbf{Y} \cup \mathbf{X})} P(w|\mathrm{Pa}(w)_G \setminus \{w\}) \prod_{\mathbf{V} \setminus (\mathbf{X} \cup \{W\})} P(v_i|\mathrm{Pa}(v_i)_G \setminus \{v_i\}) \prod_{\mathbf{U}} P(u_i).$$

Since variables in $\mathbf{U_Z}$ and $\mathbf{Z}$ can be connected to the other vertices of G only through W we can complete the marginalization over $\mathbf{Z}$ and $\mathbf{U_Z}$

$$P_{\mathbf{x}}(\mathbf{y}) = \sum_{\mathbf{U_{V \setminus Z}}} \sum_{\mathbf{V} \setminus (\mathbf{Y} \cup \mathbf{X} \cup \mathbf{Z})} P(w|\mathrm{Pa}(w)_G \setminus (\mathbf{z} \cup \{w\})) \prod_{\mathbf{V} \setminus (\mathbf{X} \cup \mathbf{Z} \cup \{W\})} P(v_i|\mathrm{Pa}(v_i)_G \setminus \{v_i\}) \prod_{\mathbf{U_{V \setminus Z}}} P(u_i).$$

Because we have assumed that $\mathbf{Z}$ is disconnected from $\mathbf{V} \setminus \mathbf{Z}$ in $G_{\overline{W}}$ we have that $G[\mathbf{V} \setminus \mathbf{Z}] = L(G, \mathbf{V} \setminus \mathbf{Z})$. Therefore, just as in the proof of Theorem 9, we have that $P(v_i|\mathrm{Pa}(v_i)_G \setminus \{v_i\}) =$

$P'(v_i|\mathrm{Pa}(v_i)_{G'} \setminus \{v_i\})$ for all $V_i \in \mathbf{V} \setminus (\mathbf{X} \cup \mathbf{Z} \cup \{W\})$ and $P(u_i) = P'(u_i)$ for all $U_i \in \mathbf{U}_{\mathbf{V}\setminus\mathbf{Z}}$. Additionally, we have $P(w|\mathrm{Pa}(w)_G \setminus (\mathbf{z} \cup \{w\})) = P(w|Pa(w)_{G'} \setminus \{w\})$. Finally we obtain

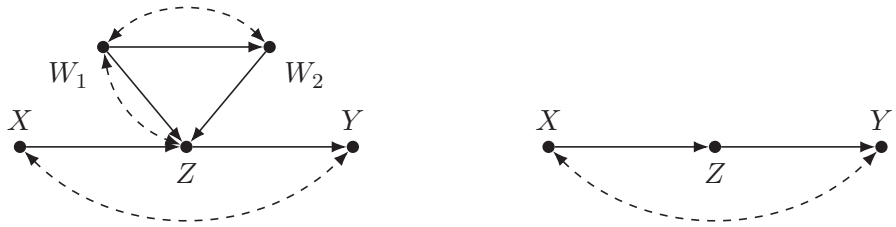
$$\begin{aligned} P_{\mathbf{x}}(\mathbf{y}) &= \sum_{\mathbf{U}_{\mathbf{V}\setminus\mathbf{Z}}} \sum_{\mathbf{V}\setminus(\mathbf{Y}\cup\mathbf{X}\cup\mathbf{Z})} P(w|\mathrm{Pa}(w)_{G'} \setminus \{w\}) \prod_{\mathbf{V}\setminus(\mathbf{X}\cup\mathbf{Z}\cup\{W\})} P(v_i|\mathrm{Pa}(v_i)_{G'} \setminus \{v_i\}) \prod_{\mathbf{U}_{\mathbf{V}\setminus\mathbf{Z}}} P(u_i) \\ &= \sum_{\mathbf{U}_{\mathbf{V}\setminus\mathbf{Z}}} \sum_{\mathbf{V}\setminus(\mathbf{Y}\cup\mathbf{X}\cup\mathbf{Z})} \prod_{\mathbf{V}\setminus(\mathbf{X}\cup\mathbf{Z})} P(v_i|\mathrm{Pa}(v_i)_{G'} \setminus \{v_i\}) \prod_{\mathbf{U}_{\mathbf{V}\setminus\mathbf{Z}}} P(u_i) \\ &= P'_{\mathbf{x}}(\mathbf{y}). \end{aligned}$$

■

Corollary 13 *Let W be a vertex of an SMG G and let $\mathbf{R} = An(W)_{G_{\overline{\mathbf{X}}}} \setminus De(\mathbf{X})_G$. When the causal effect $P_{\mathbf{x}}(\mathbf{y})$ is considered in graph G, the set of vertices $\mathbf{T} = \mathbf{R} \setminus Co(\mathbf{V} \setminus \mathbf{R})_{G_{\overline{W}}}$ can be removed from the graph if $G = G[An(\mathbf{Y})_G]$.*

Proof No descendant of $\mathbf{X}$ can be removed via theorem 12 since they are connected to $\mathbf{X}$ and the set to be removed cannot itself contain $\mathbf{X}$. By removing descendants of $\mathbf{X}$ and $\mathbf{X}$ itself, and assuming that $G = G[\mathrm{An}(\mathbf{Y})_G]$, we have that $\mathbf{T} \cap (\mathbf{X} \cup \mathbf{Y}) = \emptyset$. Thus it remains to remove those vertices from $\mathbf{R}$ that are connected to $\mathbf{V} \setminus \mathbf{R}$ through a path that does not contain W. Removal of the resulting set $\mathbf{T}$ from the graph is now licensed by theorem 12. ■

In the following example we consider the causal effect of X on Y in graph G of Figure 5(a) and show how Corollary 13 can be applied. The ID algorithm succeeds in identifying the



(a) Graph G. (b) Subgraph $G[\{X, Y, Z\}]$.

Figure 5: First example of Corollary 13.

causal effect and returns following expression for it

$$\sum_{w_1,w_2,z} P(z|x,w_1,w_2)P(w_2|x,w_1)P(w_1|x)\left(\sum_{x'} P(y|x',w_1,w_2,z)P(x')\right).$$

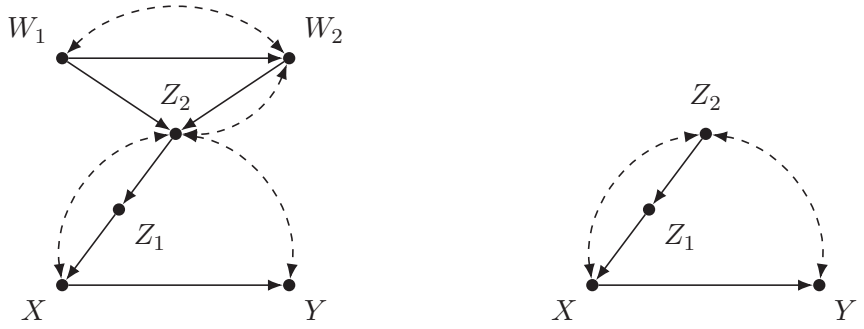
We apply Corollary 13 which allows us to remove W_1 and W_2 from the graph, since they are connected to other vertices of the graph through a single vertex Z. Applying the ID

algorithm in the resulting subgraph $G[\{X, Y, Z\}]$ of Figure 5(b) provides us the following expression

$$\sum_z P(z|x) \left(\sum_{x'} P(y|x', z) P(x') \right).$$

The same expression can be obtained manually by applying the front-door criterion.

We provide another example on Corollary 13 with a slightly more complicated graph. We are interested in the causal effect of X on Y in graph G of Figure 6(a) We obtain a



(a) Graph G. (b) Subgraph $G[\{X, Y, Z_1, Z_2\}$.

Figure 6: Second example of Corollary 13.

formula for the causal effect using the ID algorithm

$$\frac{\sum_{w_2,z_2} P(y|w_1, w_2, z_2, z_1, x) P(x|w_1, w_2, z_2, z_1) P(z_2|w_1, w_2) P(w_2|w_1)}{\left(\sum_{w_2,z_2,y} P(y|w_1, w_2, z_2, z_1, x) P(x|w_1, w_2, z_2, z_1) P(z_2|w_1, w_2) P(w_2|w_1) \right)}.$$

Vertices W_1 and W_2 are connected to other vertices only through Z_2 which allows us to remove them from the graph. We obtain a simpler formula for the causal effect from the subgraph $G[\{X, Y, Z_1, Z_2\}]$ of Figure 6(b)

$$\frac{\sum_{z_2} P(y|z_2, z_1, x) P(x|z_2, z_1) P(z_2)}{\sum_{z_2,y} P(y|z_2, z_1, x) P(x|z_2, z_1) P(z_2)}.$$

The previous results have allowed us to completely remove specific vertices from the graph. Next we will consider cases where a vertex is present in the graph, but it is not necessary to observe it. This means that instead of the original graph we may consider identifiability in the corresponding latent projection, as characterized by the following lemma.

Lemma 14 *Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be an SMG and let $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ be disjoint sets of variables. Let $P(\mathbf{V})$ be the joint distribution of $\mathbf{V}$. Then the causal effect of $\mathbf{X}$ on $\mathbf{Y}$ is identifiable from P' in the latent projection $L(G, \mathbf{V} \setminus \mathbf{Z})$ where $P' = P(\mathbf{V} \setminus \mathbf{Z})$, if and only if it is identifiable from P' in G.*

Proof Tian (2002) showed that the latent projection has the same topological relations over the observables and that it has the same set of maximal C-components. Thus if $P_{\mathbf{x}}(\mathbf{y})$

is identifiable from P' in G it is also identifiable from P' in $L(G, \mathbf{V} \setminus \mathbf{Z})$ with the same expression and vice versa. ■

In situations where $\mathbf{X}$ is a singleton we can exploit the following sufficient condition for identifiability by Tian and Pearl (2002).

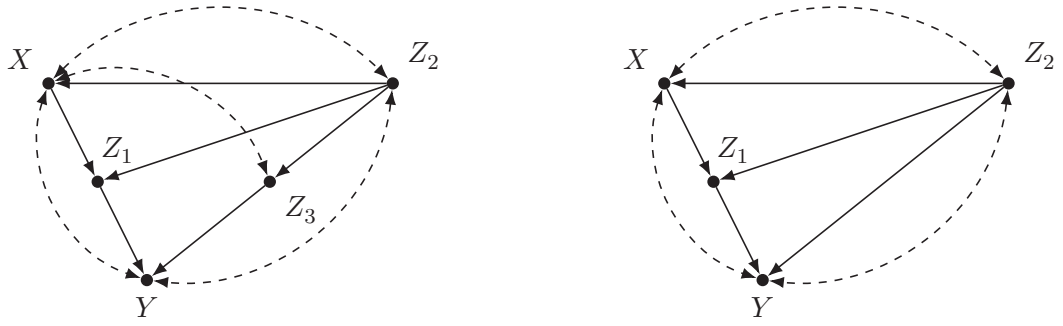
Theorem 15 *The causal effect $P_x(\mathbf{y})$ is identifiable if there is no bidirected path between X and any of its children in $G[An(\mathbf{Y})_G]$.*

We can regard any variable as latent when $\mathbf{X}$ is a singleton if the respective latent projection does not induce such a bidirected path.

Corollary 16 *Let G be an SMG and let $\mathbf{Y}$ be a set of vertices. Let $X, W \in \mathbf{V}$ such that $X \neq W$, $W \notin \mathbf{Y}$ and $X, W \in An(\mathbf{Y})_G$. The causal effect $P_x(\mathbf{y})$ obtained from P in G is equal to $P'_x(\mathbf{y})$ obtained from $P' = P(An(\mathbf{Y})_G \setminus \{w\})$ and $G' = L(G[An(\mathbf{Y})_G], An(\mathbf{Y})_G \setminus \{W\})$ if there is no bidirected path from X to any of its children in G'.*

Proof By Theorem 15 the causal effect $P'_x(\mathbf{y})$ is identifiable from P' in G'. By Lemma 14 $P'_x(\mathbf{y})$ is now identifiable from P' in G. $P_x(y)$ obtained from P' in G is equal to $P^*_x(y)$ obtained from $P^* = P(\mathrm{An}(\mathbf{Y})_G)$ in $G[\mathrm{An}(\mathbf{Y})]$ since identifiability from P' implies identifiability from P. Finally, $P^*_x(\mathbf{y})$ obtained from $P^* = P(\mathrm{An}(\mathbf{Y})_G)$ in $G[\mathrm{An}(\mathbf{Y})_G]$ is equal to $P_x(y)$ obtained from P in G. ■

We continue with an example on how Corollary 16 can be applied in practice. We consider the causal effect of X on Y in the graph G of Figure 7(a). The causal effect is



(a) Graph G. (b) Latent projection $L(G, \{X, Y, Z_1, Z_2\})$.

Figure 7: A graph for an example where Corollary 16 allows us to make variable Z_3 latent.

identifiable and the output of the ID algorithm is

$$\sum_{z_2,z_3,z_1} P(z_1|z_2,x) \frac{\left(\sum_{x'} P(y|z_2,x',z_3,z_1)P(z_3|z_2,x')P(x'|z_2)P(z_2)\right)}{\left(\sum_{x',y'} P(y'|z_2,x',z_3,z_1)P(z_3|z_2,x')P(x'|z_2)P(z_2)\right)} \times \left(\sum_{x',z_3,y'} P(y'|z_2,x',z_3,z_1)P(z_3|z_2,x')P(x'|z_2)P(z_2)\right) P(z_3|z_2).$$

We may apply Corollary 16 by noting that $G = G[\mathrm{An}(Y)_G]$ and that there is no bidirected path between X and its only child Z_1 in the latent projection $L(G, \mathbf{V} \setminus \{Z_3\})$ as depicted in Figure 7(b).

Running the ID algorithm in $L(G, \mathbf{V} \setminus \{Z_3\})$ results in the following expression

$$\sum_{z_2, z_1} \left(\sum_{x'} P(y|z_2, x', z_1) P(x'|z_2) P(z_2) \right) P(z_1|z_2, x).$$

5. Pruning Identifiability Algorithm

Corollaries 11, 13 and 16 can be implemented as additional steps for the ID algorithm. For Algorithm 2, line 3 implements Corollary 11, line 4 implements Corollary 13 and line 5 implements Corollary 16. Other lines are identical to the ID algorithm. This algorithm is provided by the R package *causaleffect* which implements various causal inference algorithms such as the original ID algorithm (Tikka and Karvanen, 2017a).

The ordering of the variables in the loop on line 4 has no effect on the resulting expression, but the ordering does matter on line 5. Choosing a different ordering may lead to a different expression. For example, when identifying the causal effect of X on Y in graph G of Figure 8 one may obtain either the back-door formula or the front-door formula by proceeding in either the topological ordering or the reverse-topological ordering of the vertices, respectively.

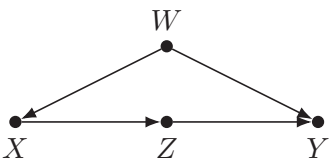


Figure 8: Graph G where different latent projections lead to different expressions for $P_x(y)$.

Algorithm 2 A pruning identifiability algorithm (PID) for causal effects.

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

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

function PID$(\mathbf{y}, \mathbf{x}, P, G)$
1: **if** $\mathbf{x} = \emptyset$,
  return $\sum_{v \in \mathbf{v} \setminus \mathbf{y}} P(\mathbf{v})$.
2: **if** $\mathbf{V} \neq \mathrm{An}(\mathbf{Y})_G$,
  return PID$(\mathbf{y}, \mathbf{x} \cap \mathrm{An}(\mathbf{y})_G, P(\mathrm{An}(\mathbf{Y})_G), G[\mathrm{An}(\mathbf{Y})_G])$
3: **let** $\mathbf{Z} = \mathrm{An}(\mathbf{Y})_G \setminus \mathrm{Co}(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$.
  if $\mathbf{Z} \neq \emptyset$ **and** $G[\mathbf{V} \setminus \mathbf{Z}] = L(G, \mathbf{V} \setminus \mathbf{Z})$,
  return PID$(\mathbf{y}, \mathbf{x} \setminus \mathbf{z}, P(\mathbf{V} \setminus \mathbf{Z}), G[\mathbf{V} \setminus \mathbf{Z}])$
4: **for** $W \in \mathbf{V} \setminus \mathbf{X}$ **do**
  let $\mathbf{R} = \mathrm{An}(W)_{G_{\overline{\mathbf{X}}}} \setminus \mathrm{De}(\mathbf{X})_G$.
  let $\mathbf{T} = \mathbf{T} \cup (\mathbf{R} \setminus \mathrm{Co}(\mathbf{V} \setminus \mathbf{R})_{G_{\overline{W}}})$.
  if $\mathbf{T} \neq \emptyset$,
  return PID$(\mathbf{y}, \mathbf{x}, P(\mathbf{V} \setminus \mathbf{T}), G[\mathbf{V} \setminus \mathbf{T}])$
5: **if** $\mathbf{X} = \{X\}$,
  let $G[\mathbf{S}_X] \in C(G), X \in \mathbf{S}_X$.
  if $\mathrm{Ch}(\mathbf{X})_{G[\mathbf{S}_X]} \setminus \mathbf{X} = \emptyset$,
    for $W \in \mathbf{V} \setminus (\mathbf{Y} \cup \mathbf{X})$ **do**
      let $G' = L(G, \mathbf{V} \setminus \{W\})$.
      let $G'[\mathbf{S}'_X] \in C(G'), X \in \mathbf{S}'_X$.
      if $\mathrm{Ch}(\mathbf{X})_{G'[\mathbf{S}'_X]} \setminus \mathbf{X} = \emptyset$,
        $P \leftarrow P(\mathbf{V} \setminus \{W\})$.
        $G \leftarrow G'$.
        $\mathbf{V} \leftarrow \mathbf{V} \setminus \{W\}$.
6: **let** $\mathbf{W} = (\mathbf{V} \setminus \mathbf{X}) \setminus \mathrm{An}(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$.
  if $\mathbf{W} \neq \emptyset$,
  return PID$(\mathbf{y}, \mathbf{x} \cup \mathbf{w}, P, G)$.
7: **if** $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}_1], \ldots, G[\mathbf{S}_k]\}$,
  return $\sum_{v \in \mathbf{v} \setminus (\mathbf{y} \cup \mathbf{x})} \prod_{i=1}^{k}$ **PID**$(\mathbf{s}_i, \mathbf{v} \setminus \mathbf{s}_i, P, G)$.
  if $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}]\}$,
8: **if** $C(G) = \{G\}$,
    throw FAIL$(G, G[\mathbf{S}])$.
9: **if** $G[\mathbf{S}] \in C(G)$,
    return $\sum_{v \in \mathbf{s} \setminus \mathbf{y}} \prod_{V_i \in \mathbf{S}} P(v_i | v_\pi^{(i-1)})$.
10: **if** $(\exists \mathbf{S}') \mathbf{S} \subset \mathbf{S}'$ **such that** $G[\mathbf{S}'] \in C(G)$,
    return PID$(\mathbf{y}, \mathbf{x} \cap \mathbf{s}', \prod_{V_i \in \mathbf{S}'} P(V_i | V_\pi^{(i-1)} \cap \mathbf{S}', v_\pi^{(i-1)} \setminus \mathbf{s}'), G[\mathbf{S}'])$.

On line 5 we first check whether the set $\mathbf{X}$ is a singleton. If so, we determine whether any children of X belong the same C-component as X. If no such children exist, we iterate over

the possible latent projections in an attempt to find one that does not induce a bidirected path between X and any of its children in the projection. After the new pruning steps have been carried out, we attempt identification using the original formulation of the ID algorithm.

We return to the example presented in the introduction and show how Algorithm 2 operates to derive the expression for $P_x(y)$ in the graph of Figure 1(a). We choose the topological ordering to be $Y > Z_1 > X > Z_3 > W_2 > Z_4 > Z_2 > W_1$. We begin on line 3, since $\mathbf{Z} = \mathrm{An}(Y)_G \setminus \mathrm{Co}(Y)_{G_{\overline{X}}} = \mathbf{V} \setminus \{W_1, W_2\}$ and continue by calling $\mathrm{PID}(y, x, P(\mathbf{V} \setminus \{W_1, W_2\}), G[\mathbf{V} \setminus \{W_1, W_2\}])$. In the presentation below, $\mathbf{V}$ and G refer to the set of vertices and graph in the current call of PID, respectively.

Next we enter the loop on line 4. When $W = Z_1$ we obtain $\mathbf{R} = \mathrm{An}(Z_1)_{G_{\overline{X}}} \setminus \mathrm{De}(X)_G = \{Z_2, Z_4\}$ and $\mathbf{R} \setminus \mathrm{Co}(\mathbf{V} \setminus \mathbf{R})_{G_{\overline{Z}_1}} = \{Z_2, Z_4\} \setminus \{Z_2\} = \{Z_4\}$ since Z_4 is an ancestor of Z_1 and it is disconnected from other vertices in $G_{\overline{Z}_1}$. Other choices of W result in an empty set. When the loop is completed we have $\mathbf{T} = \{Z_4\}$ and we continue by calling $\mathrm{PID}(y, x, P(\mathbf{V} \setminus \{Z_4\}), G[\mathbf{V} \setminus \{Z_4\}])$.

Since $\mathbf{X}$ is a singleton we end up on line 5 and find no children of X in the same C-component as X. We assume a reverse-topological ordering for the loop and begin with the latent projection $L(G, \mathbf{V} \setminus \{Z_2\})$. This projection creates a bidirected edge between X and Z_1 bringing them into the same C-component in the projection. Thus we continue with $L(G, \mathbf{V} \setminus \{Z_1\})$. This projection is also unsuitable, since Y is a child of X in the projection and there is a bidirected arc connecting them. We continue with $L(G, \mathbf{V} \setminus \{Z_3\})$ and find that X is not connected to its children via bidirected paths in the projection. Thus we set $P \leftarrow P(\mathbf{V} \setminus \{Z_3\})$, $G \leftarrow L(G, \mathbf{V} \setminus \{Z_3\})$ and $\mathbf{V} \leftarrow \mathbf{V} \setminus \{Z_3\}$. This projection is identical to the graph depicted in Figure 7(b). After these steps, only lines of the original ID algorithm are called, which results in the expression

$$\sum_{z_2, z_1} \left(\sum_{x'} P(y|z_2, z_1, x') P(x'|z_2) P(z_2) \right) P(z_1|z_2, x).$$

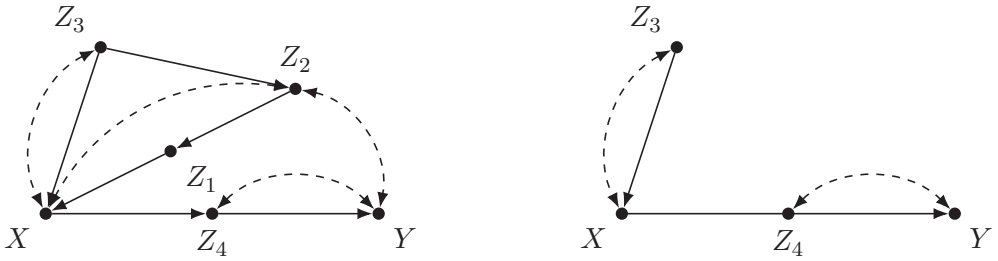
6. Examples on Recursive Pruning

Corollaries 10, 13 and 16 often provide direct benefits when applied before the ID algorithm. The following examples show why they are also useful as recursive steps as implemented in the PID algorithm.

We are tasked with identifying the causal effect of X on Y in graph G depicted in Figure 9(a)
Initially there are no vertices that are connected to other vertices only through X. As a recursive step of the ID algorithm, we are tasked with identifying $P'_{z_3,x}(y)$ from P', where

$$P' = \sum_{z_2} P(y|z_3, z_2, z_1, x) P(x|z_3, z_2, z_1) P(z_2|z_3) P(z_3),$$

in the subgraph $G[\{Z_3, X, Y\}]$ shown in Figure 9(b). In this graph Z_3 is connected to other vertices only through X and it can be removed according to Corollary 11, since the corresponding latent projection is the subgraph $G[\{X, Y\}]$. Thus we sum out Z_3 from P'



(a) Graph G. (b) Subgraph $G[\{Z_3, X, Y\}]$.

Figure 9: A graph for the example of recursive application of Corollary 11 within the ID algorithm.

and the resulting expression for the causal effect is

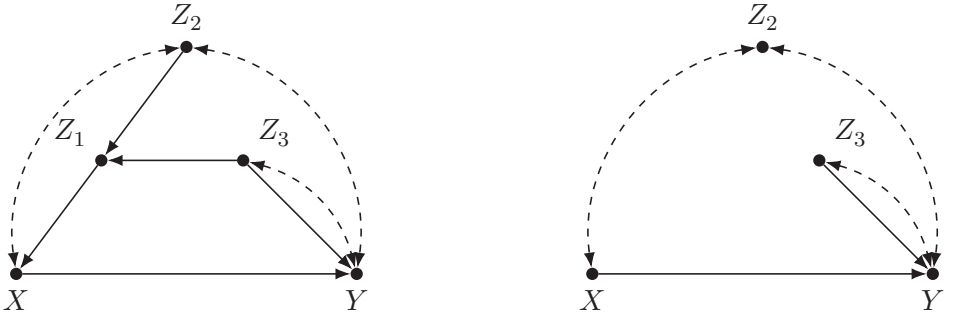
$$\sum_{z_4} \frac{\sum_{z_2,z_3} P(y|z_3,z_2,z_1,x,z_4)P(z_4|z_3,z_2,z_1,x)P(x|z_3,z_2,z_1)P(z_2|z_3)P(z_3)}{\sum_{z_2,z_3,y'} P(y'|z_3,z_2,z_1,x,z_4)P(z_4|z_3,z_2,z_1,x)P(x|z_3,z_2,z_1)P(z_2|z_3)P(z_3)}.$$

If Corollary 11 is not applied at this stage, the final expression is instead

$$\sum_{z_4} \left(\frac{\sum_{z_2} P(y|z_3,z_2,z_1,x,z_4)P(z_4|z_3,z_2,z_1,x)P(x|z_3,z_2,z_1)P(z_2|z_3)P(z_3)}{\sum_{z_2,y'} P(y'|z_3,z_2,z_1,x,z_4)P(z_4|z_3,z_2,z_1,x)P(x|z_3,z_2,z_1)P(z_2|z_3)P(z_3)} \times \right.$$
$$\left. \frac{\sum_{z_2,y'} P(y'|z_3,z_2,z_1,x,z_4)P(z_4|z_3,z_2,z_1,x)P(x|z_3,z_2,z_1)P(z_2|z_3)P(z_3)}{\sum_{z_2,y',z_4} P(y'|z_3,z_2,z_1,x,z_4)P(z_4|z_3,z_2,z_1,x)P(x|z_3,z_2,z_1)P(z_2|z_3)P(z_3)} \right).$$

Using Corollary 11 provides us with a simpler expression in this situation by completely removing the second term from the product inside the summation.

Next we show how Corollary 13 can also be applied at a recursive step. Our interest lies in the causal effect of X on Y in graph G of Figure 10(a)



(a) Graph G. (b) Subgraph $G[\{Z_2, Z_3, X, Y\}]$.

Figure 10: A graph for the example of recursive application of Corollary 13 within the ID algorithm.

There are no vertices that are connected to other vertices in the graph via a single vertex. When the ID algorithm is applied we eventually reach a step where the causal effect of

$P'_{z_2,x}(y)$ is to be identified from P', where

$$P' = P(y|z_2, z_3, z_1, x)P(x|z_2, z_3, z_1)P(z_3|z_2)P(z_2),$$

in the subgraph $G[\{Z_2, Z_3, X, Y\}]$ which is depicted in Figure 10(b).

In this graph Z_3 can be removed according to Corollary 13, since Z_3 is connected to other vertices of the subgraph only through Y. The resulting distribution is obtained by summing out Z_3 from P'. The final expression for the causal effect is now

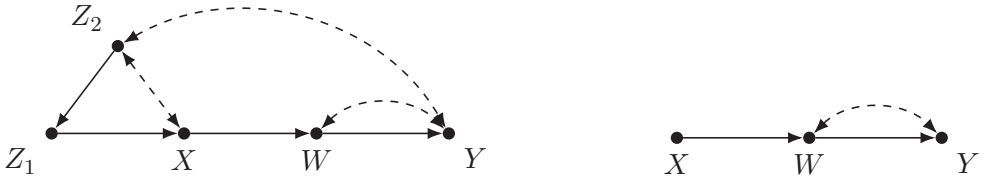
$$\frac{\sum_{z_2,z_3} P(y|z_2, z_3, z_1, x)P(x|z_2, z_3, z_1)P(z_3|z_2)P(z_2)}{\sum_{z_2,z_3,y'} P(y'|z_2, z_3, z_1, x)P(x|z_2, z_3, z_1)P(z_3|z_2)P(z_2)}. \tag{1}$$

If Corollary 13 is not applied, the resulting expression is instead

$$\sum_{z_3} \left(\frac{\sum_{z_2} P(y|z_2, z_3, z_1, x)P(x|z_2, z_3, z_1)P(z_3|z_2)P(z_2)}{\sum_{z_2,y'} P(y'|z_2, z_3, z_1, x)P(x|z_2, z_3, z_1)P(z_3|z_2)P(z_2)} \times \sum_{z_2,x,y'} P(y'|z_2, z_3, z_1, x)P(x|z_2, z_3, z_1)P(z_3|z_2)P(z_2) \right).$$

As in the previous example, the benefit of applying Corollary 13 is apparent.

We can also take advantage of latent projections recursively via Corollary 16 as shown in the next example. Our interest lies in the causal effect of X on Y in graph G of Figure 11.



(a) Graph G. (b) Subgraph $G[\{X, W, Y\}]$.

Figure 11: A graph for the example of recursive application of Corollary 16 within the ID algorithm.

Here X is connected to its child W via a bidirected path and thus they belong to the same C-component rendering Corollary 16 unusable at this time. However, as a recursive step of the ID algorithm we have to identify $P'_x(y)$ from P', where

$$P' = \sum_{z_2} P(y|z_2, z_1, x, w)P(w|z_2, z_1, x)P(x|z_2, z_1)P(z_2)$$

in the subgraph $G[\{X, W, Y\}]$. In this subgraph X is not connected to its child W via a bidirected path. We also find that the latent projection $L(G[\{X, W, Y\}], \{X, Y\})$ does not induce such a path between X and Y. We can now continue identification in this latent projection and sum out W from P'. The resulting expression for the causal effect is

$$\frac{\sum_{z_2,w} P(y|z_2, z_1, x, w)P(w|z_2, z_1, x)P(x|z_2, z_1)P(z_2)}{\sum_{z_2,w,y'} P(y'|z_2, z_1, x, w)P(w|z_2, z_1, x)P(x|z_2, z_1)P(z_2)},$$

whereas the expression without applying the corollary is instead

$$\sum_w \left(\frac{\sum_{z_2} P(y|z_2,z_1,x,w)P(w|z_2,z_1,x)P(x|z_2,z_1)P(z_2)}{\sum_{z_2,y'} P(y'|z_2,z_1,x,w)P(w|z_2,z_1,x)P(x|z_2,z_1)P(z_2)} \times \right.$$
$$\left. \frac{\sum_{z_2,y'} P(y'|z_2,z_1,x,w)P(w|z_2,z_1,x)P(x|z_2,z_1)P(z_2)}{\sum_{z_2,w',y'} P(y'|z_2,z_1,x,w)P(w'|z_2,z_1,x)P(x|z_2,z_1)P(z_2)} \right).$$

Additional examples are provided as an R script (R Core Team, 2017) at the JMLR online paper repository. The script also includes all of the examples presented in this paper.

An interesting question is how pruning works together with simplification presented in (Tikka and Karvanen, 2017b). We return to the example on identifying the causal effect of X on Y in the graph of Figure 10. If we apply the ID algorithm without pruning and perform simplification as a post-processing step, then the resulting expression is

$$\sum_{z_3} \frac{\sum_{z_2} P(y|z_2,z_3,z_1,x)P(x|z_2,z_3,z_1)P(z_3|z_2)P(z_2)}{\sum_{z_2} P(x|z_2,z_3,z_1)P(z_3|z_2)P(z_2)} P(z_3|z_2). \tag{2}$$

This expression is in some aspects simpler than expression (1) obtained using pruning alone, but does contain a sum over Z_3 that was not originally present.

When pruning is introduced to the ID algorithm and simplification is again applied, the resulting expression is instead

$$\frac{\sum_{z_3,z_2} P(y|z_2,z_3,z_1,x)P(x|z_2,z_3,z_1)P(z_3|z_2)P(z_2)}{\sum_{z_2} P(x|z_2,z_1)P(z_2)},$$

which is noticeably simpler than expressions (1) and (2). This example shows, that when pruning methods are employed together with simplification, a simpler expression can be reached than what is possible with either pruning or simplification alone.

7. Discussion

We have presented criteria for removing variables from causal models that are not necessary to achieve identifiability for a given causal effect, and showed how these criteria can be applied in practice. We integrated our results into a new version of the ID algorithm called PID as presented in Algorithm 2 to facilitate automatic processing of identifiability queries. It should be noted that the ID algorithm already performs some pruning such as removing non-ancestors of $\mathbf{Y}$.

The pruning operations carried out by Algorithm 2 can significantly simplify the resulting expression compared to the traditional ID algorithm. Benefits of simplification can be realized in various settings. A simpler expression is easier to understand and to evaluate, since the dimensionality of the problem has been reduced. This is especially true for settings where the expression has to be evaluated repeatedly. Simplification can also help dealing with data where some variables are affected by bias or contain missing data. Obtaining an expression that does not contain these variables has clear advantages. It may also be of interest to obtain a different expression for the same causal effect.

The choice of variable ordering on line 5 of Algorithm 2 is not arbitrary. However, available external knowledge may guide our selection to prefer certain orderings. For example, in a situation where two latent projections are mutually exclusive, we may prefer an ordering where the variable that is associated with the smallest cost, or of which we have the most accurate measurements is not considered latent.

When PID is applied in conjunction with simplification methods described in (Tikka and Karvanen, 2017b), various situations that lead to complex expressions can be taken into account. These methods complement each other, since the results in this paper deal with completely removing variables from the resulting expression, whereas the simplification methods focus on symbolic summation of so-called atomic expressions, which are expression consisting of a single sum and a number of product terms. An expression for a causal effect may consist of multiple atomic expressions, some of which can be simplified and some of which can not.

We showed via examples that our improvements are not simply pre-processing steps to be carried out before calling the ID algorithm, but actually provide significant benefits when applied recursively. As the ID algorithm manipulates the original graph it often enables the application of our results as well. As hedges characterize identifiability, it is possible to consider latent projections in a more general manner, but this is not necessarily beneficial for simplification. One could construct an algorithm that performs a search over the possible subsets of $\mathbf{V}$, and checks whether identifiability is retained in the corresponding latent projection. However, as we have shown via examples, this may not be enough to obtain a simpler expression, and the recursive structure of the ID algorithm needs to be taken advantage of. Instead, we could consider a variant of the PID algorithm, where line 5 is replaced by this procedure. However, one must be careful when applying this method, so that the computation does not become intractable when the number of vertices increases due to the complexity of the search.

Acknowledgments

We wish to thank Professor Jukka Nyblom for his comments that greatly helped to improve this paper. We also thank the anonymous reviewers for their insightful feedback. The work belongs to the profiling area "Decision analytics utilizing causal models and multiobjective optimization" (DEMO) supported by Academy of Finland (grant number 311877).

References

A. P. Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society B*, 41:1–31, 1979.

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.

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

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

J. Pearl. Causal diagrams for empirical research. *Biometrika*, 82:669–688, 1995.

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

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2017. URL `https://www.R-project.org/`.

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, 2006.

I. Shpitser and J. Pearl. Complete identification methods for the causal hierarchy. *Journal of Machine Learning Research*, 9:1941–1979, 2008. ISSN 1532-4435.

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

J. Tian. *Studies in Causal Reasoning and Learning*. PhD thesis, Department of Computer Science, University of California, Los Angeles, 2002.

J. Tian and J. Pearl. A general identification condition for causal effects. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 567–573. AAAI Press, 2002.

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`.

T. S. Verma. Graphical aspects of causal models. Technical report, Department of Computer Science, University of California, Los Angeles, 1993. R-191.

134. TÖRMÄKANGAS, TIMO, Simulation study on the properties of quantitative trait model estimation in twin study design of normally distributed and discrete event-time phenotype variables. (417 pp.) 2012
135. ZHANG, GUO, Liouville theorems for stationary flows of generalized Newtonian fluids. (14 pp.) 2012
136. RAJALA, TUOMAS, Use of secondary structures in the analysis of spatial point patterns. (27 pp.) 2012
137. LAUKKARINEN, EIJA, On Malliavin calculus and approximation of stochastic integrals for Lévy processes. (21 pp.) 2012
138. GUO, CHANGYU, Generalized quasidisks and the associated John domains. (17 pp.) 2013
139. ÄKKINEN, TUOMO, Mappings of finite distortion: Radial limits and boundary behavior. (14 pp.) 2014
140. ILMAVIRTA, JOONAS, On the broken ray transform. (37 pp.) 2014
141. MIETTINEN, JARI, On statistical properties of blind source separation methods based on joint diagonalization. (37 pp.) 2014
142. TENGVALL, VILLE, Mappings of finite distortion: Mappings in the Sobolev space $W^{1,n-1}$ with integrable inner distortion. (22 pp.) 2014
143. BENEDICT, SITA, Hardy-Orlicz spaces of quasiconformal mappings and conformal densities. (16 pp.) 2014
144. OJALA, TUOMO, Thin and fat sets: Geometry of doubling measures in metric spaces. (19 pp.) 2014
145. KARAK, NIJJWAL, Applications of chaining, Poincaré and pointwise decay of measures. (14 pp.) 2014
146. JYLHÄ, HEIKKI, On generalizations of Evans and Gangbo's approximation method and L^∞ transport. (20 pp.) 2014
147. KAURANEN, AAPO, Space-filling, energy and moduli of continuity. (16 pp.) 2015
148. YLINEN, JUHA, Decoupling on the Wiener space and variational estimates for BSDEs. (45 pp.) 2015
149. KIRSILÄ, VILLE, Mappings of finite distortion on generalized manifolds. (14 pp.) 2015
150. XIANG, CHANG-LIN, Asymptotic behaviors of solutions to quasilinear elliptic equations with Hardy potential. (20 pp.) 2015
151. ROSSI, EINO, Local structure of fractal sets: tangents and dimension. (16 pp.) 2015
152. HELSKE, JOUNI, Prediction and interpolation of time series by state space models. (28 pp.) 2015
153. REINIKAINEN, JAAKKO, Efficient design and modeling strategies for follow-up studies with time-varying covariates. (36 pp.) 2015
154. NUUTINEN, JUHO, Maximal operators and capacities in metric spaces. (22 pp.) 2016
155. BRANDER, TOMMI, Calderón's problem for p-Laplace type equations. (21 pp.) 2016
156. ÄRJE, JOHANNA, Improving statistical classification methods and ecological status assessment for river macroinvertebrates. (30 pp.) 2016
157. HELSKE, SATU, Statistical analysis of life sequence data. (40 pp.) 2016
158. RUOSTEENOJA, EERO, Regularity properties of tug-of-war games and normalized equations. (16 pp.) 2017
159. ZHANG, YI, Planar Sobolev extension domains. (13 pp.) 2017
160. YLITALO, ANNA-KAISA, Statistical inference for eye movement sequences using spatial and spatio-temporal point processes. (45 pp.) 2017
161. LINDQVIST, PETER, Notes on the p-Laplace equation. (106 pp.) 2017
162. LEHTONEN, JERE, Injectivity results for the geodesic ray transform. (15 pp.) 2017
163. GOLO, SEBASTIANO NICOLUSSI, Topics in the geometry of non-Riemannian Lie groups. (14 pp.) 2017
164. KOPRA, JUHO, Statistical modelling of selective non-participation in health examination surveys. (25 pp.) 2018
165. HEINO, JOONAS, On the local and global regularity of tug-of-war games. (20 pp.) 2018
166. MUKHERJEE, SHIRSO, Regularity of quasilinear sub-elliptic equations in the Heisenberg group. (18 pp.) 2018
167. NANDI, DEBANJAN, Applications of the quasihyperbolic metric. (14 pp.) 2018

ISBN 978-951-39-7518-0 (print)
ISBN 978-951-39-7519-7 (pdf)
ISSN 1457-8905