

UCLA

UCLA Electronic Theses and Dissertations

Title

Empowering Causal Inference with Parametric Constraints

Permalink

<https://escholarship.org/uc/item/8mx411zs>

ISBN

9798247901082

Author

Chen, Yizuo

Publication Date

2026-05-21

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Empowering Causal Inference with Parametric Constraints

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Yizuo Chen

2026

© Copyright by

Yizuo Chen

2026

ABSTRACT OF THE DISSERTATION

Empowering Causal Inference with Parametric Constraints

by

Yizuo Chen

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2026

Professor Adnan Youssef Darwiche, Chair

Understanding and reasoning about causal relationships is central to modeling the world and making intelligent decisions. At the heart of causal inference lies the identifiability problem, which asks whether a causal query can be uniquely determined from the available information. This dissertation demonstrates the benefits of leveraging additional background knowledge — beyond causal graphs and observational data — to improve the identifiability of causal effects, a widely studied class of causal queries. In particular, we present methods that incorporate background knowledge in the form of parametric constraints on the identifiability problem, including functional dependencies (e.g., legal driving age is a deterministic function of country) and context-specific independencies (e.g., when it is warm outside, whether the heater is on becomes independent of the number of occupants), enabling the identification of causal effects that would otherwise be impossible in the classical setup. We also propose a systematic approach that integrates various forms of constraints into identifiability using tractable arithmetic circuits. Beyond identifiability, we investigate the impact of knowledge, such as functional dependencies, on accelerating the computation of causal queries when causal models are fully specified. This gives rise to the notion of

causal treewidth, which establishes tighter complexity bounds for computing marginals and applies directly to queries such as causal effects. We further extend these computational improvements to more sophisticated counterfactual reasoning by bounding its complexity in terms of causal treewidth.

The dissertation of Yizuo Chen is approved.

Chad Hazlett

Aditya Grover

Yizhou Sun

Adnan Youssef Darwiche, Committee Chair

University of California, Los Angeles

2026

To my family

TABLE OF CONTENTS

1	Introduction	1
2	Causal Effect Identifiability with Parametric Constraints	6
2.1	Causal Bayesian Networks and Interventions	6
2.2	Causal Effect Identifiability	8
2.3	Incorporating Knowledge Into Causal Effect Identifiability	9
3	Identifying Causal Effects Under Functional Dependencies	12
3.1	Functional Dependencies	12
3.2	Interplay with Positivity Constraints	14
3.3	Functional Elimination and Projection	17
3.4	Treating Hidden Functional Variables	21
3.5	Treating Observed Functional Variables	23
3.6	A Different Approach Under Weaker Positivity Assumptions	26
3.7	Experiments	27
4	Identifying Causal Effects Under Context-Specific Independencies	29
4.1	Two Levels of Causal Effect Identifiability	29
4.2	The Impact of CSI Constraints on Identifiability	31
4.3	The Impact of State Constraints on Identifiability	33
4.4	Testing Identifiability Under CSI and State Constraints	35
4.4.1	Leveraging CSI Constraints	35
4.4.2	Leveraging State Constraints	42

4.4.3	Heuristics	44
4.5	Experiments	45
5	Testing Constrained Identifiability Using Tractable Arithmetic Circuits	48
5.1	Background on Tractable Arithmetic Circuits (ACs)	48
5.2	Procedure for Testing Identifiability Using ACs	49
5.2.1	AC Construction	50
5.2.2	Invariance Testing	51
5.3	Finding Invariant-Cuts For ACs	54
5.4	Exploiting Constraints in Identifiability: Examples	56
5.4.1	Context-Specific Independencies	57
5.4.2	Fully-Known Observational Distributions	58
6	Accelerating Computations with Functional Dependencies	63
6.1	Computing Marginals	64
6.1.1	Factor Operations	64
6.1.2	Jointrees and Treewidth	65
6.2	Exploiting Qualitative Functional Dependencies	67
6.3	Thinned Jointrees: Definition and Properties	71
6.3.1	Jointree Thinning	71
6.3.2	Mechanism Replication	75
6.4	Experiments	77
7	Extensions to Counterfactual Reasoning	81
7.1	Counterfactual Reasoning in Fully Specified SCMs	82

7.2	Bounding the Causal Treewidth of Twin Networks	84
7.3	Counterfactual Reasoning Beyond Two Worlds	87
7.4	Beyond Fully Specified SCMs	89
7.5	Experiments	90
8	Conclusion	93
A	Appendix for Chapter 2	95
A.1	Projection	95
A.2	Background on C-Components	96
A.3	Causal Effect Identification Algorithms	97
B	Appendix for Chapter 3	98
B.1	Functional Elimination for Causal Bayesian Networks	98
B.2	Proofs	100
C	Appendix for Chapter 4	119
C.1	Properties of Context-Induced Graphs	119
C.2	Heuristics	120
C.3	Proofs	122
C.4	Complete Experimental Results	139
D	Appendix for Chapter 5	142
D.1	Proofs	142
E	Appendix for Chapter 6	146
E.1	Variable Elimination Under Functional Dependencies	146

E.2 Proofs	147
F Appendix for Chapter 7	160
F.1 Proofs	160
References	163

LIST OF FIGURES

1.1	An illustration showing that an unidentifiable causal effect may become identifiable with additional knowledge (parametric constraints).	2
2.1	Mutilated and projected graphs of a causal graph. Hidden variables are indicated with boxes. A bidirected edge ($V_1 \longleftrightarrow V_2$) is a compact notation for $V_1 \leftarrow H \rightarrow V_2$, where H is an auxiliary hidden variable. Figure 2.1(d) shows an example of CPT for D	7
2.2	A Venn diagram illustrating the model sets considered for identifiability under various levels of knowledge (constraints). The outermost circle represents classical identifiability without any constraints, and the constraint set $\mathcal{C}_1$ is a subset of $\mathcal{C}_2$	10
3.1	Examples for positivity.	15
3.2	Contrasting projection with functional projection. C, D are functional. Hidden variables are indicated with boxes. Figure 3.2(b) is the result of projecting (a) on observed variables. Figure 3.2(c) is the result of functionally eliminating C, D from (a). Figure 3.2(d) is the result of functional projection with functional variables C, D	20
3.3	B is functional.	21
3.4	Variables A, B, C, F, X, Y are observed. Variables D, E are functional.	22
3.5	$B_1, \dots, B_n$ are functional.	24
4.1	Causal graphs showing the effectiveness of CSI constraints, with hidden variables shown in rectangles.	33
4.2	Causal graphs showing the effectiveness of state constraints when coupled with CSIs.	34

4.3	A causal graph, distinct context-induced graphs, and a mutilated graph under the context b_0c_0	38
4.4	Context-induced and mutilated graphs for Figure 4.1.	39
4.5	CI-graphs under different contexts in the context-partition for various examples.	43
4.6	Number of identifiable causal effects (out of 200 runs) as the number of variables increases with $r = 0.8$. Methods are labeled with the suffix “_var” for variable-based identifiability and “_state” for state-based identifiability. The two levels of identifiability are equivalent for CLASSICAL-ID.	45
5.1	Arithmetic circuits for two different models.	49
5.2	AC constructed for $\Pr_x(y)$ for a causal graph; only one branch for each $+$ -node is plotted; equivalent expressions are shown in blue and nodes in the invariant-cut are marked with red boxes.	52
5.3	A causal graph from [TP02b]. Hidden variables are indicated with boxes.	54
5.4	An AC constructed for $\Pr_x(y)$; an invariant-cut (in red boxes) is found by the c-component method.	55
5.5	Causal graphs that exhibit different behaviors of identifiability for $\Pr_x(y)$	57
5.6	AC for Figure 5.5(a) with projections derived with the c-component method; projections are marked near the nodes in blue; the invariant-cut is marked with red boxes.	58
5.7	CSI example from Chapter 4 and an AC with an invariant-cut for $\Pr_x(y)$	59
5.8	Models for $\Pr(\mathbf{V})$	60
5.9	Examples for fully-known $\Pr(\mathbf{V})$	61
5.10	ACs constructed for Figure 5.9(a) and Figure 5.9(c) under known $\Pr^*(\mathbf{V})$	61
6.1	A BN, a jointree, and the jointree with its separators and clusters.	65

6.2	A CPT f appears next to a jointree node i iff f is hosted by i . The mechanism for the functional variable D is replicated in (c), where red variables are thinned.	68
6.3	A Bayesian network, its renamed replication, a jointree and a thinned jointree for the replication. Variables B and C in the BN are functional. Red variables are thinned.	69
6.4	A Bayesian network with functional variables B and C	73
6.5	Two jointree thinnings. Each edge (i, j) is marked by separator $\mathbf{S}_{ij}$. Red variables are not in the thinned separator $\mathbf{S}_{ij}^*$	73
6.6	A BN with an exponentially sized complete replication.	76
6.7	Comparing the thinned jointrees constructed using Theorem 11 with the ones in [Dar20]. Functional variables are restricted to be non-root.	78
6.8	Illustrating the impact of replication strategies.	79
7.1	SCM with endogenous variables A, B, C and exogenous variables U_1, U_2 . All variables are binary. The mechanisms for endogenous variables are specified by structural equations.	82
7.2	Computing counterfactual $\Pr(a_s \bar{a}, \bar{s})$ using a twin network. Variables with evidence or interventions are indicated with boxes.	84
7.3	Illustrating the construction of a twin jointree from a base jointree.	86
7.4	A 3-world network where the roots A, B are duplicated.	88
7.5	Width of jointrees (y-axis) against number of variables in the base network (x-axis) for maximum number of parents $p = 5$	91
A.1	The causal graph G , its mutilated graph under $\Pr_F(I)$, and the c-component decomposition. Hidden variables are shown in rectangles, and variables in different c-components are highlighted with different colors.	96

B.1	Variables X, Y_1, Z, Y_2 are observed. Variable Z is functional.	111
C.1	CI-graph constructed for the causal effect $\Pr_x(y)$ and CSI constraint $(Z \perp\!\!\!\perp U \mid x)$.	124
C.2	Context-induced and mutilated graphs for the examples in Figure 4.1 under the CSI $(J \perp\!\!\!\perp D \mid A, Y = 0)$	131
C.3	Mutilated graphs for the examples in Figure 4.5.	135
C.4	CI-graph of Figure 4.2(b) under the context $B=1$	137

LIST OF TABLES

3.1	A CPT for B and a functional CPT for C	13
3.2	Numbers of causal effects that are unidentifiable (UID) and that become identifiable under functional dependencies (UID-FID), and the average number of observed variables passed to project-ID ($\#OBS$), for causal graphs with varying numbers of variables (N) and functional variables (W).	28
6.1	Illustrating factors and their key operations, multiplication and sum-out.	64
6.2	Average size of replications. Replication nc means: the number of replicas for a node X is between $(n - 1)c$ and nc where c is the number of children for node X	79
C.1	We record the number of causal effects deemed identifiable for each method under causal graphs with varying numbers of nodes (n) and CSIs (r). We also record the number of runs that complete within the 5-minute timeout (labeled as “success” in the table). For runs that exceed the timeout, we set the runtime to 5 minutes when computing the average.	141

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Adnan Darwiche, for his constant support and guidance since my first year of undergraduate study. Professor Darwiche has been involved in nearly all my projects and patiently taught me research skills from the ground up, from formulating problems to presenting research. His passion for exploration, along with his deep insights, has shaped me into a better researcher and helped me stay focused on what matters most: the joy of discovery.

I would like to thank the other members of my doctoral committee, Chad Hazlett, Yizhou Sun, and Aditya Grover, for their time and insightful feedback.

I have greatly benefited from my fellow group members Arthur Choi, Haiying Huang, Jason Shen, and Yunqiu Han for their collaboration and discussions, as well as from Scott Mueller, Ilya Shpitser, and Jin Tian for their valuable feedback on my work. I would like to thank my industry mentors Jane Barker (Amazon) and Amit Bhatia (RTX) for making my internships both enjoyable and rewarding. I am also grateful to Jie Qiao from RIT, who introduced me to my first research project on laser simulation and trained me to think rigorously while I was in high school.

I want to thank my friends who have enriched my life outside of research: Qian and Jason for tennis; Hanwen for concerts and bouldering; Haiying for board games and hiking; Xinta for the trip to Joshua Tree; Gwen and Zoe for cat care advice; Oliver, Renato, Daniel, and Benjie for squash; and Poorva for keeping me updated on the latest.

Finally, I am deeply grateful to my parents and my brother Yiyu for their unconditional love and encouragement, without which none of this would have been possible, and to my cat Eighteen for keeping me company through many long nights.

VITA

2018-2021 B.S. in Computer Science, Minor in Mathematics, University of California,
Los Angeles

PUBLICATIONS

Yizuo Chen and Adnan Darwiche. “On the Definition and Computation of Causal Treewidth”
In UAI, Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence, 2022.

Yunqiu Han, Yizuo Chen, and Adnan Darwiche. “On the Complexity of Counterfactual
Reasoning.” In IJCAI, pp. 5676–5684. ijcai.org, 2023.

David Huber, Yizuo Chen, Alessandro Antonucci, Adnan Darwiche, and Marco Zaffalon.
“Tractable Bounding of Counterfactual Queries by Knowledge Compilation.” Sixth Work-
shop on Tractable Probabilistic Modeling at UAI, 2023.

Yizuo Chen and Adnan Darwiche. “Identifying Causal Effects Under Functional Dependen-
cies.” *Entropy*, 26(12), 2024.

Yizuo Chen, Haiying Huang, and Adnan Darwiche. “Towards an effective practice of learning
from data and knowledge.” *Int. J. Approx. Reason.*, 171:109188, 2024.

Yizuo Chen and Adnan Darwiche. “Constrained Identifiability of Causal Effects.” AAAI
Workshop on Artificial Intelligence with Causal Techniques, 2025.

Yizuo Chen and Adnan Darwiche. “On the Granularity of Causal Effect Identifiability.”
arXiv, abs/2510.16703, 2025.

CHAPTER 1

Introduction

Understanding causality is essential for uncovering how the world operates. While correlations between events can support predictions based on observed patterns, causal relationships are crucial for making informed decisions that achieve reliable outcomes. For example, knowing that car accidents are often caused by speeding leads to the placement of speed limit signs. In contrast, a common belief that car brands are a major factor in accidents turns out to be causally unfounded; i.e., restricting certain car brands would have little to no effect on overall accident rates.

The study of causation has led to questions (or queries) of varying levels of sophistication, which have been categorized into separate layers within the causal hierarchy [pea18]. One popular class of causal queries is *causal effect*, which quantifies how interventions influence outcomes of interest, exemplified by the question, “What is the probability that a patient would recover if a doctor prescribed a drug?” Here, the doctor’s prescription represents an intervention, as it actively changes the patient’s behavior, which is fundamentally different from observing patients taking drugs naturally (often suggesting that the patient is experiencing severe symptoms). In fact, causal effects belong to the second layer of the causal hierarchy and ultimately require experimental studies, such as randomized controlled trials, to answer. One drawback of conducting experiments, however, is that they can be expensive or even infeasible in practice. For instance, conducting an experiment on climate change may require manipulating the environmental conditions (e.g., CO₂ level) in ecosystems, which can be extremely costly. Experiments that test the effect of smoking on lung

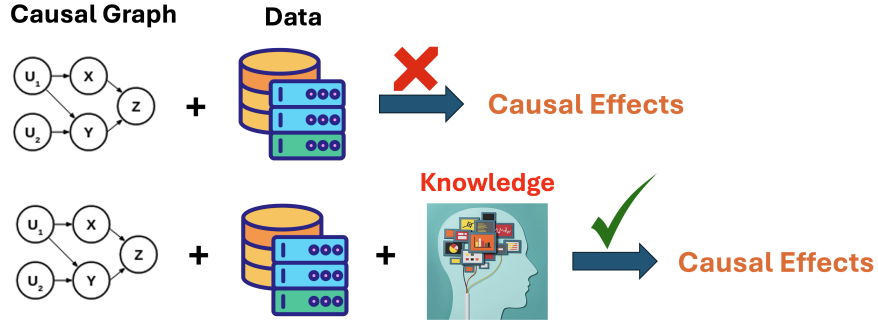


Figure 1.1: An illustration showing that an unidentifiable causal effect may become identifiable with additional knowledge (parametric constraints).

cancer may require participants to smoke cigarettes, which is clearly unethical.

A major research direction in causal inference aims to avoid such experiments by estimating causal effects with the aid of causal graphs, which are essentially directed acyclic graphs (DAGs) in which each edge represents a cause-and-effect relationship. A common setup is to combine these causal graphs with (partial) observational data to compute causal effects without conducting experiments. This has led to the *identifiability* problem, which asks whether the value of a causal effect can be uniquely determined from the available information, which includes a causal graph and a distribution over a subset of variables (available from observational data) in the common setup. In the past few decades, a significant amount of effort has been devoted to studying the identifiability of causal effects; see, e.g., [Pea95, pea09, SGS00, IR15, PJS17, HR20]. Some early works include the back-door criterion [Pea93, pea09] and the front-door criterion [Pea95, pea09]. These criteria are sound but incomplete, as they may fail to identify certain causal effects that are, indeed, identifiable. Complete identification methods include the do-calculus method [pea09], the identification algorithm introduced in [TP03, HV06], and the ID algorithm proposed in [SP06].

One main limitation of these algorithms is their strong reliance on the observability of variables. When data are collected on only a limited set of variables, many causal effects remain unidentifiable, as the available data do not provide sufficient information to determine their values. This situation arises, for example, when most variables (such as gender and

age) involve confidential or sensitive information and are therefore unobserved. A main goal of this dissertation is to address this issue by incorporating additional *background knowledge* (parametric constraints) into the identification of causal effects, providing information beyond causal graphs and observational data.¹ We demonstrate that causal effects unidentifiable in the classical setup can become identifiable given such additional knowledge. To this end, we formulate various types of knowledge — including functional dependencies, context-specific independencies, and known parameters — as additional constraints on the identifiability problem, leading to the notion of *constrained identifiability* [CD24, CD25a]. The study of constrained identifiability further empowers a more granular type of identifiability that pertains to a *particular state* of treatment on a *particular state* of outcome, in contrast to the classical setup where treatment and outcome are defined in terms of variables. Interestingly, these two levels of identifiability (state-based and variable-based) are identical in the classical setting but become distinct in the presence of additional knowledge, such as context-specific independencies. We demonstrate this separation with examples and present sound algorithms for testing both levels of identifiability.

The second part of this dissertation focuses on a different setting in which the available information consists of a fully specified model; that is, both a causal graph and its full parameterization are known. Under this setup, all causal effects are identifiable so the identifiability problem is no longer of interest. Instead, we study the problem of computing causal queries, which requires conducting marginal inference on the causal model and is known to be $\#P$ -hard [Coo90, KC15]. In this dissertation, we address this challenge by exploiting the knowledge of functional dependencies, enabling speedups in marginal inference and, consequently, more efficient computation of queries such as causal effects. We do so by introducing an algorithmic approach for incorporating available functional dependencies and by formalizing the notion of *causal treewidth* to provide a tighter bound on inference

¹We will use the terms “background knowledge” and “parametric constraints” interchangeably throughout this dissertation. In general, background knowledge is broader and may also include information beyond parametric constraints, such as graph structure.

complexity. Importantly, such speedups can be achieved without the need to know the quantitative details of the functional dependencies, making the approach applicable in settings where parameters are learned from data or are subject to change.

The benefits of leveraging functional dependencies can be further extended to improve the efficiency of *counterfactual reasoning*, which corresponds to the third layer of the causal hierarchy and involves more sophisticated queries with conflicting variable values. To illustrate, a counterfactual query may ask, “What is the probability that a patient would have recovered had they taken the drug, given that they did not take the medicine and died?” To answer such queries, it is standard to consider *structural causal models (SCMs)*, which by definition encode abundant functional dependencies (all non-root variables in an SCM are assumed to be functional) [pea09]. Hence, it is conceivable that the methods that exploit functional dependencies can be leveraged to accelerate the computation of counterfactual queries, assuming the SCMs are fully specified. This further yields a complexity bound for counterfactual reasoning in terms of causal treewidth.

This dissertation is structured as follows.

In Chapter 2, we review classical causal effect identifiability and extend it to incorporate additional background knowledge, leading to the notion of *constrained identifiability*.

In Chapter 3, we incorporate the knowledge of functional dependencies into constrained identifiability. This is achieved through a novel graph-based operation that removes variables from the graph, enabling a larger class of causal effects to be identified. We also examine the interplay between functional dependencies and positivity constraints.

In Chapter 4, we define a more fine-grained notion of identifiability that involves particular states of treatments and outcomes, and illustrate its properties through examples. We further propose a sound method for testing identifiability under context-specific independencies (CSIs). We also discuss the impact of constraints that restrict variable states.

In Chapter 5, we introduce a systematic framework for testing constrained identifiability

using tractable Arithmetic Circuits (ACs). We use examples to demonstrate the effectiveness of this AC-based approach by showing that previously unidentifiable causal effects may become identifiable under various combinations of constraints.

In Chapter 6, we present an approach to improve the computational efficiency of queries (associational and interventional) in fully specified models by leveraging the knowledge of functional dependencies. We formalize the data structure of *thinned jointrees* for exploiting functional dependencies and formally define *causal treewidth* as a complexity bound for marginal inference (query computation) using thinned jointrees.

In Chapter 7, we extend the previous results to accelerate counterfactual reasoning, establishing a complexity bound in terms of causal treewidth and showing that it is no harder than associational and interventional reasoning when causal models are fully specified. The results also apply to general queries involving multiple counterfactual worlds and changing states of exogenous variables.

We close the dissertation with concluding remarks in Chapter 8. All proofs are provided in the appendix.

CHAPTER 2

Causal Effect Identifiability with Parametric Constraints

We start by reviewing, in Section 2.1, an important class of causal models known as Causal Bayesian Networks (CBNs), which are commonly used for computing causal effects [pea09]. The operation of *intervention* can then be represented as a modification on the CBN, providing a basis for formally defining causal effects and the identifiability problem, which we review in Section 2.2. We then show in Section 2.3 that background knowledge can be incorporated as additional constraints on the identifiability problem, leading to a formal definition of *constrained identifiability*. The results in this chapter appeared in [CD24, CD25a].

2.1 Causal Bayesian Networks and Interventions

Throughout the dissertation, we assume all variables are discrete. Single variables are denoted by uppercase letters (e.g. X) and their states are denoted by lowercase letters (e.g. x). Sets of variables are denoted by bold uppercase letters (e.g. $\mathbf{X}$) and their instantiations are denoted by bold lowercase letters (e.g. $\mathbf{x}$).

A Causal Bayesian Network (CBN) is a pair $\langle G, \Theta \rangle$, where G is a *causal graph* in the form of a directed acyclic graph (DAG) and Θ is a set of conditional probability tables (CPTs). We have one CPT for each variable X with parents $\mathbf{P}$ in G , denoted as $\theta_{X|\mathbf{P}}$, which specifies the conditional probability distributions $\Pr(X|\mathbf{P})$. For example, Figure 2.1(a) depicts the causal graph, and Figure 2.1(d) shows the corresponding CPT for D in the CBN. It follows

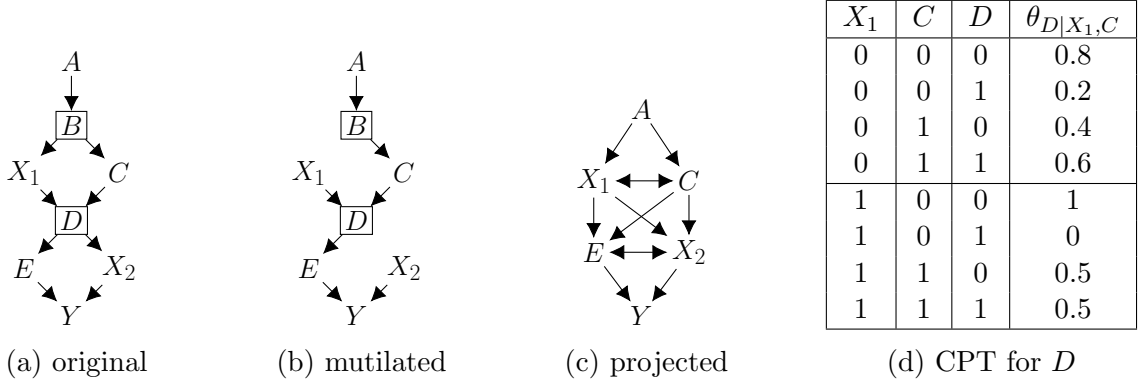


Figure 2.1: Mutilated and projected graphs of a causal graph. Hidden variables are indicated with boxes. A bidirected edge ($V_1 \longleftrightarrow V_2$) is a compact notation for $V_1 \leftarrow H \rightarrow V_2$, where H is an auxiliary hidden variable. Figure 2.1(d) shows an example of CPT for D .

that every CPT $\theta_{X|\mathbf{P}}$ satisfies the following properties: $\theta_{x|\mathbf{p}} \in [0, 1]$ for all instantiations $(x, \mathbf{p})$ and $\sum_x \theta_{x|\mathbf{p}} = 1$ for each instantiation $\mathbf{p}$. Since CPTs are a special type of factors that satisfy the above probability properties, we may also denote them as $f_X(X, \mathbf{P})$ or f_X .

A CBN induces a joint distribution over its variables $\mathbf{V}$ as the product of its CPTs, i.e., $\Pr(\mathbf{V}) = \prod_{V \in \mathbf{V}} \theta_{V|\mathbf{P}}$, where $\mathbf{P}$ denotes the parents of V . In the CBN shown in Figure 2.1(a), for example, the joint probability can be computed as

$$\Pr(A, B, X_1, C, D, E, X_2, Y) = \theta_A \cdot \theta_{B|A} \cdot \theta_{X_1|B} \cdot \theta_{C|B} \cdot \theta_{D|X_1,C} \cdot \theta_{E|D} \cdot \theta_{X_2|D} \cdot \theta_{Y|E,X_2}$$

Applying an *intervention* (or treatment) $do(\mathbf{x})$ forces the variables $\mathbf{X}$ to take the values $\mathbf{x}$. For example, setting car's speed to 25 corresponds to the intervention $do(\text{Speed} = 25)$. The intervention yields a new distribution called the *interventional distribution*, denoted by $\Pr_{\mathbf{x}}(\mathbf{V})$. The standard way to compute the interventional distribution is to consider the *mutilated CBN* $\langle G', \Theta' \rangle$ that is constructed from the original CBN $\langle G, \Theta \rangle$ as follows: Remove from G all edges that point to variables in $\mathbf{X}$; then, replace the CPT in Θ for each $X \in \mathbf{X}$ with a CPT θ'_X where $\theta'_x = 1$ if x is consistent with $\mathbf{x}$ and $\theta'_x = 0$ otherwise. Figure 2.1(b) depicts the mutilated causal graph G' of the graph in Figure 2.1(a) under a treatment $do(x_1, x_2)$. The interventional distribution $\Pr_{\mathbf{x}}(\mathbf{V})$ is the distribution induced by the mutilated CBN

$\langle G', \Theta' \rangle$, and the causal effect $\Pr(\mathbf{Y}|do(\mathbf{x}))$ can be computed as the marginal $\Pr_{\mathbf{x}}(\mathbf{Y})$ in the mutilated CBN. In this example, the causal effect of $do(x_1, x_2)$ on Y can be computed by

$$\Pr_{x_1 x_2}(y) = \sum_{abcde} \theta_a \cdot \theta_{b|a} \cdot \theta_{c|b} \cdot \theta_{d|x_1, c} \cdot \theta_{e|d} \cdot \theta_{y|e, x_2}$$

2.2 Causal Effect Identifiability

A key question in causal inference is to check whether a causal effect can be (uniquely) computed given the causal graph G and a distribution $\Pr(\mathbf{V})$ over a subset $\mathbf{V}$ of its variables. If the answer is yes, we say that the causal effect is *identifiable* from G and $\Pr(\mathbf{V})$. Otherwise, the causal effect is *unidentifiable*. Variables $\mathbf{V}$ are said to be *observed*, and the remaining variables are said to be *hidden*, where $\Pr(\mathbf{V})$ is often estimated from observational data collected on $\mathbf{V}$. We start with the general definition of causal effect identifiability from [TP02a, pea09], with a slight rephrasing. Here, we use “model” to mean a full parametrization of the causal graph (i.e., a fully specified CBN).

Definition 1 (Causal Effect Identifiability). *The causal effect of $\mathbf{X}$ on $\mathbf{Y}$, denoted $\Pr_{\mathbf{x}}(\mathbf{Y})$, is identifiable wrt $\langle G, \mathbf{V} \rangle$ if $\Pr_{\mathbf{x}}^1(\mathbf{y}) = \Pr_{\mathbf{x}}^2(\mathbf{y})$ for all instantiations $\mathbf{x}, \mathbf{y}$ and any pair of models that induce $\Pr^1, \Pr^2$ such that $\Pr^1(\mathbf{V}) = \Pr^2(\mathbf{V})$.*

That is, the problem of identifiability checks whether every pair of models that induces the same distribution $\Pr(\mathbf{V})$ also produces the same value for the causal effect. Equivalently, to show that a causal effect is unidentifiable, it suffices to find two CBNs that induce the same distribution $\Pr(\mathbf{V})$ yet different causal effects. Note that Definition 1 does not restrict the considered models based on the properties of the distributions they induce. However, in the literature on identifying causal effects, it is quite common to only consider models that induce distributions that satisfy some positivity constraints, such as $\Pr(\mathbf{V}) > 0$ in existing identification methods including IDENTIFY [TP03] and the do-calculus [pea09]. We examine such constraints more carefully in Chapter 3, as they may contradict knowledge such as

functional dependencies, which we introduce later.

It is well known that under some positivity constraints (e.g., $\Pr(\mathbf{V}) > 0$), the identifiability of causal effects can be efficiently tested using a two-step procedure, which we call the *project-ID* algorithm. Given a causal graph G , the procedure first applies the projection operation proposed in [Ver93, TP02b, TP03] to yield a new causal graph G' whose hidden variables are all roots, each with exactly two children. The second step then applies identification algorithms such as ID [SP06] or IDENTIFY [TP03] to G' to check the identifiability of the causal effect. These algorithms are sound and complete: they yield an identifying formula if the causal effect is identifiable and resulting in an outcome of FAIL otherwise.

Consider the causal effect $\Pr_{X_1 X_2}(Y)$ in Figure 2.1(a), where hidden variables are the non-root variables $\{B, D\}$. We first project the causal graph G in Figure 2.1(a) onto its observed variables to yield the causal graph G' in Figure 2.1(c) (all hidden variables in G' are auxiliary and roots). We then run the ID algorithm on G' , which returns the following (simplified) identifying formula: $\Pr_{x_1 x_2}(y) = \sum_c \Pr(c) \sum_e \Pr(y|e, x_2) \Pr(e|x_1, c)$. Hence, the causal effect is identifiable and can be computed using the above formula. Moreover, all quantities in the formula can be obtained from the distribution $\Pr(A, C, E, X_1, X_2, Y)$ over observed variables, which can be estimated from observational data. More details on the projection operation and the identification algorithms can be found in Appendix A.

2.3 Incorporating Knowledge Into Causal Effect Identifiability

When background knowledge (constraints) is available, we can generalize the classical identifiability setup by incorporating it as another input to the identifiability problem, alongside the causal graph and observed variables. The benefit of knowledge has been previously investigated in [THK19], which showed that certain unidentifiable causal effects can become identifiable given information about context-specific independence. Our research follows the same direction but considers additional types of knowledge, such as functional dependencies

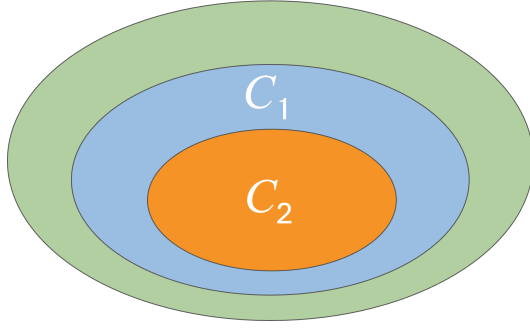


Figure 2.2: A Venn diagram illustrating the model sets considered for identifiability under various levels of knowledge (constraints). The outermost circle represents classical identifiability without any constraints, and the constraint set $\mathcal{C}_1$ is a subset of $\mathcal{C}_2$.

and known parameters. In this section, we focus on the general implications of background knowledge on the identifiability problem, leaving the detailed methods for leveraging each type of knowledge to later chapters.

Recall that in the setup of classical identifiability in Definition 1, we do not restrict the models (parameterization) assigned to the causal graph, and hence all possible models are considered when testing identifiability. This, however, is no longer the case when additional knowledge is available. To illustrate, consider the causal graph in Figure 2.1(a). If we know that variable D functionally depends on its parents (i.e., D is deterministic), the CPT shown in Figure 2.1(d) cannot be assigned to D in the models since it contains non-deterministic values. Similarly, if the knowledge specifies that Y is independent of X_2 when $E = 0$, the models cannot define Y as a disjunction of E and X_2 . These examples illustrate that background knowledge restricts the set of models that can be assigned to a causal graph.

We first define the (constrained) identifiability tuple, which includes the constraints $\mathcal{C}$ as an additional input, along with the causal graph and the set of observed variables considered in classical identifiability.

Definition 2. We call $\langle G, \mathbf{V}, \mathcal{C} \rangle$ an identifiability tuple when G is a causal graph, $\mathbf{V}$ is its set of observed variables, and $\mathcal{C}$ is a set of constraints on the models for G .

The notion of constrained identifiability can now be defined as follows. For simplicity,

we will say “identifiability” to mean “constrained identifiability” when the context is clear.

Definition 3. Let $\langle G, \mathbf{V}, \mathcal{C} \rangle$ be an identifiability tuple. A causal effect $\Pr_{\mathbf{X}}(\mathbf{Y})$ is (constrained) identifiable wrt $\langle G, \mathbf{V}, \mathcal{C} \rangle$ if $\Pr_{\mathbf{x}}^1(\mathbf{y}) = \Pr_{\mathbf{x}}^2(\mathbf{y})$ for all instantiations $\mathbf{x}, \mathbf{y}$ and any pair of models that satisfy $\mathcal{C}$ and induce distributions $\Pr^1, \Pr^2$ where $\Pr^1(\mathbf{V}) = \Pr^2(\mathbf{V})$.

In the example above, we can write the constraints as $\mathcal{C} = \{\mathbf{W} = \{D\}, (Y \perp\!\!\!\perp X_2 \mid E = 0)\}$, where $\mathbf{W}$ denotes variables with functional dependencies. When $\mathcal{C}$ contains a single constraint, we may omit the curly braces for convenience; for example, we may write $\langle G, \mathbf{V}, \Pr(\mathbf{V}) > 0 \rangle$ instead of $\langle G, \mathbf{V}, \{\Pr(\mathbf{V}) > 0\} \rangle$. When no constraint is imposed, i.e., $\mathcal{C} = \emptyset$, we obtain the weakest version of identifiability in which all models are considered.

CHAPTER 3

Identifying Causal Effects Under Functional Dependencies

We consider the constraints of functional dependencies and develop methods for incorporating them into the identifiability problem. We begin with a formal definition of functional dependencies in Section 3.1 and investigate their interplay with positivity constraints in Section 3.2. In Section 3.3, we present an elimination operation that removes functional variables from a causal graph while preserving key properties of the resulting causal graph, including the identifiability of causal effects. In Sections 3.4 and 3.5, we demonstrate some key improvements enabled by this elimination scheme. First, an unidentifiable causal effect may become identifiable when certain variables are functional. Second, certain functional variables can be excluded from being observed without affecting the identifiability of a causal effect, which may significantly reduce the number of needed variables in observational data. In addition to such elimination-based approach, we present an alternative method for testing identifiability that works under weaker positivity assumptions in Section 3.6. We report some empirical results in Section 3.7. The results in this chapter appeared in [CD24].

3.1 Functional Dependencies

A variable X in a causal graph is said to functionally depend on its parents $\mathbf{P}$ if its CPT is deterministic: $\theta_{x|\mathbf{p}} \in \{0, 1\}$ for every instantiation $x, \mathbf{p}$. Variable X is also said to be *functional* in this case. In this work, we assume *qualitative* functional dependencies: we do

A	B	C	$\theta_{B A}$	$\theta_{C A}$
0	0	0	0.2	0
0	1	1	0.8	1
1	0	0	0.6	1
1	1	1	0.4	0

Table 3.1: A CPT for B and a functional CPT for C .

not know the parameter $\theta_{X|\mathbf{P}}$ (or $\Pr(X|\mathbf{P})$), we only know that it is deterministic.¹

Table 3.1 shows two variables B and C that both have A as their parent. Variable C is functional but variable B is not. The CPT for variable C will be called a *functional CPT* in this case. Functional CPTs are also known as (causal) mechanisms and are often expressed using structural equations in structural causal models (SCMs) [BP95, GP98, Hal00]. For example, the CPT for C in Table 3.1 can be represented using the equation $C = \neg A$.

Qualitative functional dependencies are a longstanding concept. For example, they are common in relational databases, see, e.g., [HM08, Dat12], and their relevance to probabilistic reasoning had been brought up early in [Pea88, Ch. 3]. One example of a (qualitative) functional dependency is that different countries have different driving ages, so we know that “Driving Age” functionally depends on “Country” even though we may not know the specific driving age for each country. Another example is that a “Letter Grade” (e.g., B+) for a class is functionally dependent on the student’s “Weighted Average” (e.g., 87) even though we may not know the scheme for converting a weighted average to a letter grade.

In this work, we assume that we are given a causal graph G in which some variables $\mathbf{W}$ have been designated as functional. As illustrated in Chapter 2, the presence of functional variables imposes constraints that restrict the set of models considered when checking identifiability. We denote such constraints introduced by functional dependencies as $\mathcal{C} = \{\mathbf{W}\}$.

¹We assume that root variables are not functional as such variables can be removed from the causal graph.

3.2 Interplay with Positivity Constraints

Functional dependencies may violate the positivity commonly assumed in the literature on causal effect identifiability. For example, if variable A is a parent of functional variable B , then the positivity assumption $\Pr(A, B) > 0$ cannot hold. To examine the interplay between functional dependencies and positivity, we formalize the notion of positivity constraints.

Definition 4. A positivity constraint on $\Pr(\mathbf{V})$ is an inequality of the form $\Pr(\mathbf{S}|\mathbf{Z}) > 0$ where $\mathbf{S} \subseteq \mathbf{V}$, $\mathbf{Z} \subseteq \mathbf{V}$ and $\mathbf{S} \cap \mathbf{Z} = \emptyset$. That is, for all instantiations $\mathbf{s}, \mathbf{z}$, if $\Pr(\mathbf{z}) > 0$ then $\Pr(\mathbf{s}, \mathbf{z}) > 0$.

When $\mathbf{Z} = \emptyset$, the positivity constraint is defined on a marginal distribution, $\Pr(\mathbf{S}) > 0$. To illustrate, the positivity constraint $\Pr(X_2|B, X_1, C) > 0$ specifies the constraint that $\Pr(x_2, b, x_1, c) > 0$ if $\Pr(b, x_1, c) > 0$ for every instantiation x_2, b, x_1, c . We may impose multiple positivity constraints on the observed variables $\mathbf{V}$. We will use $\mathcal{C}_{\mathbf{V}}$ to denote the set of positivity constraints imposed on $\Pr(\mathbf{V})$ and use $\text{vars}(\mathcal{C}_{\mathbf{V}})$ to denote all the variables mentioned by $\mathcal{C}_{\mathbf{V}}$. Consider the constraints $\mathcal{C}_{\mathbf{V}} = \{\Pr(X_1, X_2|B, C) > 0, \Pr(Y|C, D) > 0\}$, then $\text{vars}(\mathcal{C}_{\mathbf{V}}) = \{X_1, X_2, B, C, D, Y\}$. The weakest set of positivity constraints is $\mathcal{C}_{\mathbf{V}} = \emptyset$ (no positivity constraints), and the strongest positivity constraint is $\mathcal{C}_{\mathbf{V}} = \{\Pr(\mathbf{V}) > 0\}$ (strict positivity). The strict positivity is widely assumed in existing causal effect identification algorithms, including the IDENTIFY algorithm [TP03] and do-calculus [pea09].

Since positivity constraints restrict the set of models for the causal graph, they naturally fit into the notion of constrained identifiability in Definition 3. Specifically, we can write the identifiability tuple as $\langle G, \mathbf{V}, \mathcal{C}_{\mathbf{V}} \rangle$. When coupled with functional dependencies on variables $\mathbf{W}$, we write the tuple as $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$, indicating that the models considered must both contain the functional variables $\mathbf{W}$ and induce distributions that satisfy the positivity $\mathcal{C}_{\mathbf{V}}$.²

²This definition is equivalent to the “functional identifiability” in [CD24]; we represent functional dependencies $\mathbf{W}$ as a constraint in $\mathcal{C}$ to maintain consistency with other chapters.



Figure 3.1: Examples for positivity.

To further stress the importance of positivity constraints, we show that without positivity constraints, most causal effects would not be identifiable. To achieve this, we define a notion called *first ancestor* on a causal graph as follows. A treatment $X \in \mathbf{X}$ is said to be a first ancestor of some outcome $Y \in \mathbf{Y}$ if X is an ancestor of Y in causal graph G , and there exists a directed path from X to Y that is not intercepted by $\mathbf{X} \setminus \{X\}$. Consider the causal graph in Figure 3.1(a) with hidden variable U . The treatment X_2 is a first ancestor of outcome Y_2 , and outcome Y_1 does not have any first ancestor. In general, there must exist a first ancestor if some treatment variable is an ancestor of some outcome variable in the causal graph. The following result states a criterion under which a causal effect is never identifiable.

Proposition 1. *The casual effect of $\mathbf{X}$ on $\mathbf{Y}$ is not identifiable wrt $\langle G, \mathbf{V}, \mathcal{C}_{\mathbf{V}} \rangle$ if some $X \in \mathbf{X}$ is a first ancestor of some $Y \in \mathbf{Y}$, and $\mathcal{C}_{\mathbf{V}}$ does not imply $\Pr(X) > 0$.*

Hence, identifiability is not possible without some positivity constraints if at least one treatment variable is an ancestor of some outcome variable (which is common). By Proposition 1, the causal effect of $\{X_1, X_2\}$ on $\{Y_1, Y_2\}$ is not identifiable in Figure 3.1(a) if the considered distributions do not satisfy $\Pr(X_2) > 0$.

As positivity constraints become stronger, causal effects are more likely to be identifiable since the set of considered models becomes smaller. That is, an unidentifiable causal effect under positivity constraints $\mathcal{C}_{\mathbf{V}}^1$ may become identifiable under positivity constraints $\mathcal{C}_{\mathbf{V}}^2$ if $\mathcal{C}_{\mathbf{V}}^2$ implies $\mathcal{C}_{\mathbf{V}}^1$. Consider the causal graph in Figure 3.1(b) in which all variables are observed, $\mathbf{V} = \{X, Y, Z\}$. Without positivity constraints, i.e., $\mathcal{C}_{\mathbf{V}} = \emptyset$, the causal effect of X on Y is unidentifiable. However, it becomes identifiable given strict positivity, i.e., $\mathcal{C}_{\mathbf{V}} =$

$\{\Pr(X, Y, Z) > 0\}$, leading to the identifying formula $\Pr_x(y) = \sum_z \Pr(y|x, z) \Pr(z)$. This causal effect is also identifiable under the weaker positivity constraint $\mathcal{C}_{\mathbf{V}} = \{\Pr(X|Z) > 0\}$,³ which implies $\Pr(X) > 0$, so strict positivity is not necessary for identifiability even though it is typically assumed for this folklore result. This is an example where strict positivity may be assumed for technical convenience only as it may facilitate the application of some identifiability techniques like the do-calculus [pea09].

Both $\mathcal{C}_{\mathbf{V}}$ and $\mathbf{W}$ represent constraints on the models when checking identifiability, and these two types of constraints may contradict each other. We next define notions that characterize some important interactions between positivity and functional constraints.

Definition 5. *Consider an identifiability tuple $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$. Then $\mathcal{C}_{\mathbf{V}}$ and $\mathbf{W}$ are called consistent if there exists a model for G which induces a distribution satisfying $\mathcal{C}_{\mathbf{V}}$ and in which $\mathbf{W}$ are functional. Moreover, $\mathcal{C}_{\mathbf{V}}$ and $\mathbf{W}$ are called separable if $\mathbf{W} \cap \text{vars}(\mathcal{C}_{\mathbf{V}}) = \emptyset$.*

If $\mathcal{C}_{\mathbf{V}}$ is inconsistent with $\mathbf{W}$, the set of distributions $\Pr$ considered in Definition 3 is empty and, hence, the causal effect is not well defined (and trivially identifiable). As such, one would usually want to ensure such consistency. Here are some examples of positivity constraints that are always consistent with a set of functional variables $\mathbf{W}$: positivity on each treatment variable, i.e., $\{\Pr(X) > 0, X \in \mathbf{X}\}$, positivity on the set of non-functional treatments, i.e., $\{\Pr(\mathbf{X} \setminus \mathbf{W}) > 0\}$, positivity on all non-functional variables, i.e., $\{\Pr(\mathbf{V} \setminus \mathbf{W}) > 0\}$. It turns out that all these examples are special cases of the following condition. For a functional variable $W \in \mathbf{W}$, let $\mathbf{H}_W$ be variables that intercept all directed paths from non-functional variables to W (such $\mathbf{H}_W$ may not be unique). If none of the positivity constraint in $\mathcal{C}_{\mathbf{V}}$ mentions both W and $\mathbf{H}_W$, then $\mathcal{C}_{\mathbf{V}}$ and $\mathbf{W}$ are guaranteed to be consistent; see Proposition 12 in Appendix B.2.

³In this example, the positivity assumption $\mathcal{C}_{\mathbf{V}} = \{\Pr(X|Z) > 0\}$ is sufficient to make the identifying formula well-defined. This is because $\Pr(y|x, z) \Pr(z)$ in the formula is equal to zero when $\Pr(z) = 0$, and is computable when $\Pr(z) > 0$ (the conditional probability $\Pr(y|x, z)$ is well-defined if $\Pr(x|z) > 0$).

Separability is a stronger condition and it intuitively implies that the positivity constraints will not rule out any possible functions for the variables in $\mathbf{W}$. Some examples of positivity constraints that are separable from $\mathbf{W}$ are $\{\Pr(\mathbf{X} \setminus \mathbf{W}) > 0\}$ and $\{\Pr(\mathbf{V} \setminus \mathbf{W}) > 0\}$. Studying the interactions between positivity constraints and functional variables, as we did in this section, will prove helpful later when utilizing existing identification algorithms (which require positivity constraints) for testing identifiability under functional dependencies.

3.3 Functional Elimination and Projection

Our main approach for testing identifiability under functional dependencies will be based on eliminating functional variables from the causal graph, which may be followed by invoking the project-ID algorithm on the resulting graph. This can be subtle though since the described process will not work for every functional variable as we discuss later. Moreover, one needs to handle the interaction between positivity constraints and functional variables carefully. The first step though is to formalize the process of eliminating a functional variable and to study the associated guarantees.

Eliminating variables from a probabilistic model is a well studied operation, also known as marginalization; see, e.g., [ZP96, Dec96, Dar09]. When eliminating variable X from a model that represents distribution $\Pr(\mathbf{Z})$, the goal is to obtain a model that represents the marginal distribution $\Pr(\mathbf{Y}) = \sum_x \Pr(x, \mathbf{Y})$ where $\mathbf{Y} = \mathbf{Z} \setminus \{X\}$. Elimination can also be applied to a DAG G that represents conditional independencies $\mathcal{I}$, leading to a new DAG G' that represents independencies $\mathcal{I}'$ that are implied by $\mathcal{I}$. In fact, the projection operation of [Ver93, TP02b] we discussed earlier can be understood in these terms. We next propose an operation that eliminates functional variables from a DAG and that comes with stronger guarantees compared to earlier elimination operations as far as preserving independencies.

Definition 6. *The functional elimination of a variable X from a DAG G yields a new DAG attained by adding an edge from each parent of X to each child of X and then removing X*

from G .⁴

For convenience, we sometimes say “elimination” to mean “functional elimination” when the context is clear. From the viewpoint of independence relations, functional elimination is not sound if the eliminated variable is not functional. In particular, the DAG G' resulting from this elimination process may satisfy independencies (identified by d-separation) that do not hold in the original DAG G . As we show later, however, every independence implied by G' must be implied by G if the eliminated variable is functional. In the context of SCMs, functional elimination may be interpreted as replacing the eliminated variable X by its function in all structural equations that contain X . Functional elimination applies in broader contexts than SCMs though.

Functional elimination preserves independencies that hold in the original DAG and which are not preserved by other elimination methods including projection as defined in [Ver93, TP02b]. These independencies are captured using the notion of D-separation [GP88, GVP90] which is more refined than the classical notion of d-separation [Pea86, VP88] (uppercase D- versus lowercase d-). The original definition of D-separation can be found in [GVP90]. We provide a simpler definition next, stated as Proposition 2, as the equivalence between the two definitions is not immediate.

Proposition 2. *Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be disjoint variable sets and $\mathbf{W}$ be a set of functional variables in DAG G . Then $\mathbf{X}$ and $\mathbf{Y}$ are D-separated by $\mathbf{Z}$ in $\langle G, \mathbf{W} \rangle$ iff $\mathbf{X}$ and $\mathbf{Y}$ are d-separated by $\mathbf{Z}'$ in G where $\mathbf{Z}'$ is obtained as follows. Initially, $\mathbf{Z}' = \mathbf{Z}$. Repeat the next step until $\mathbf{Z}'$ stops changing: add to $\mathbf{Z}'$ every variable in $\mathbf{W}$ whose parents are in $\mathbf{Z}'$.*

To illustrate the difference between d-separation and D-separation, consider the DAG in Figure 3.2(a) and assume that variables C, D are functional. Variable G and I are not d-separated by A but they are D-separated by A . That is, there are distributions which are induced by the DAG in Figure 3.2(a) and in which G and I are not independent given

⁴Appendix B.1 extends this definition to CBNs (i.e., updating both CPTs and causal graph).

A. However, G and I are independent given A in every induced distribution in which the variables C, D are functionally determined by their parents. Functional elimination preserves D-separation in the following sense.

Theorem 1. *Consider a DAG G with functional variables $\mathbf{W}$. Let G' be the result of functionally eliminating variables $\mathbf{W}' \subseteq \mathbf{W}$ from G . For any disjoint sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ in G' , $\mathbf{X}$ and $\mathbf{Y}$ are D-separated by $\mathbf{Z}$ in $\langle G, \mathbf{W} \rangle$ iff $\mathbf{X}$ and $\mathbf{Y}$ are D-separated by $\mathbf{Z}$ in $\langle G', \mathbf{W} \setminus \mathbf{W}' \rangle$.*

The above result is stated with respect to eliminating a subset of the functional variables. If we eliminate all functional variables, then D-separation is reduced to d-separation. For example, variables G and I are D-separated by A in Figure 3.2(c) and also in Figure 3.2(a) as suggested by Theorem 1. In fact, G and I are also d-separated by A in Figure 3.2(c) since we eliminated all functional variables. we get the following stronger result.

Corollary 1. *Consider a DAG G with functional variables $\mathbf{W}$. Let G' be the result of functionally eliminating all variables $\mathbf{W}$ from G . For any disjoint sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ in G' , $\mathbf{X}$ and $\mathbf{Y}$ are d-separated by $\mathbf{Z}$ in G' iff $\mathbf{X}$ and $\mathbf{Y}$ are D-separated by $\mathbf{Z}$ in $\langle G, \mathbf{W} \rangle$.*

We now define the operation of functional projection which augments the original projection operation in [Ver93, TP02b] in the presence of functional dependencies.

Definition 7. *Let G be a DAG, $\mathbf{V}$ be its observed variables, and $\mathbf{W}$ be its hidden functional variables ($\mathbf{W} \cap \mathbf{V} = \emptyset$). The functional projection of G on $\mathbf{V}$ is a DAG obtained by functionally eliminating variables $\mathbf{W}$ from G then projecting the resulting DAG on $\mathbf{V}$.*

We will now contrast functional projection and classical projection using the causal graph in Figure 3.2(a), assuming that the observed variables are $\mathbf{V} = \{A, B, G, H, I\}$ and the functional variables are $\mathbf{W} = \{C, D\}$. Applying classical projection to this causal graph yields the causal graph in Figure 3.2(b). To apply functional projection, we first functionally eliminate C, D from Figure 3.2(a), which yields Figure 3.2(c), then project Figure 3.2(c) on variables $\mathbf{V}$ which yields the causal graph in Figure 3.2(d). If we contrast Figure 3.2(b)

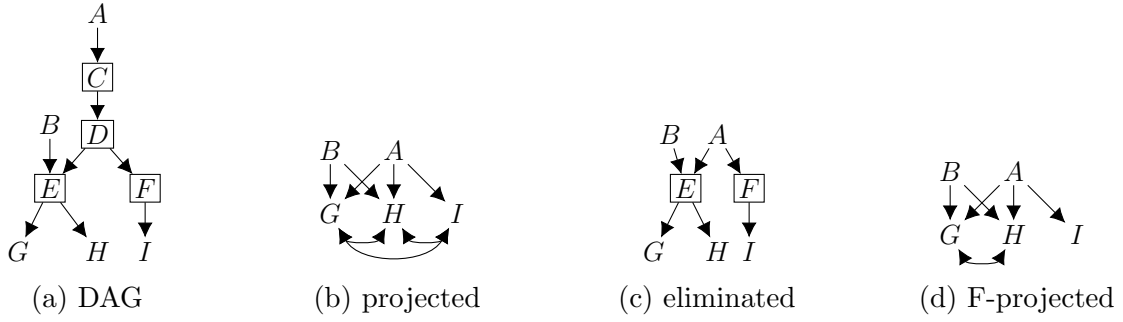


Figure 3.2: Contrasting projection with functional projection. C, D are functional. Hidden variables are indicated with boxes. Figure 3.2(b) is the result of projecting (a) on observed variables. Figure 3.2(c) is the result of functionally eliminating C, D from (a). Figure 3.2(d) is the result of functional projection with functional variables C, D .

(classical projection) with Figure 3.2(d) (functional projection), the latter is a strict subset of the former as it is missing two bidirected edges. One implication of this is that variables G and I are not d-separated by A in Figure 3.2(b) because they are not d-separated in Figure 3.2(a). However, they are D-separated in Figure 3.2(a) and hence they are d-separated in Figure 3.2(d). So functional projection yielded a DAG that exhibits more independencies. Again, this is because G and I are D-separated by A in the original DAG, a fact that is not visible to projection but is visible to functional projection.

An important corollary of functional projection is the following.

Corollary 2. *Let G be a DAG, $\mathbf{V}$ be its observed variables, $\mathbf{W}$ be its functional variables which are all hidden, and let G' be the result of functionally projecting G on $\mathbf{V}$. For any disjoint sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ in G' , $\mathbf{X}$ and $\mathbf{Y}$ are d-separated by $\mathbf{Z}$ in G' iff $\mathbf{X}$ and $\mathbf{Y}$ are D-separated by $\mathbf{Z}$ in $\langle G, \mathbf{W} \rangle$.*

In other words, classical projection preserves d-separation but functional projection preserves D-separation which subsumes d-separation. Corollary 2 is a bit more subtle and powerful than it may first seem. First, it concerns D-separations based on hidden functional variables, not all functional variables. Second, it shows that such D-separations in G appear as classical d-separations in G' which allows us to feed G' into existing identifiability

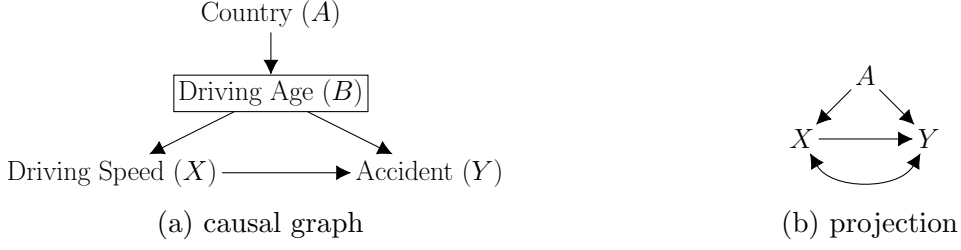


Figure 3.3: B is functional.

algorithms as we show later. This is a key enabler of some results we shall present next on testing identifiability under functional dependencies.

3.4 Treating Hidden Functional Variables

We start with an example. Consider the causal graph G in Figure 3.3(a) which models the causes of car accidents. Let $\mathbf{V} = \{A, X, Y\}$ be its observed variables. According to Definition 3 of identifiability, the causal effect of X on Y is not identifiable with respect to $\langle G, \mathbf{V}, \mathcal{C}_{\mathbf{V}} \rangle$ where $\mathcal{C}_{\mathbf{V}} = \{\Pr(A, X, Y) > 0\}$. We can show this by projecting the causal graph G on the observed variables $\mathbf{V}$, which yields the causal graph G' in Figure 3.3(b), then applying the ID algorithm [SP06] to G' which returns FAIL. Suppose now that the hidden variable B is known to be functional. This additional knowledge reduces the number of considered models so it actually renders the causal effect identifiable — the identifying formula is $\Pr_x(y) = \sum_a \Pr(a) \Pr(y|a, x)$ as we show later. Hence, an unidentifiable causal effect became identifiable in light of knowledge that some variable is functional even without knowing the quantity of the function.

The question now is: How do we algorithmically test identifiability when some variables are functional? We will propose two techniques for this purpose, the first of which is geared towards exploiting existing algorithms for classical identifiability. This technique is based on eliminating functional variables from the causal graph while preserving identifiability, with the goal of getting to a point where none of the remaining variables are functional

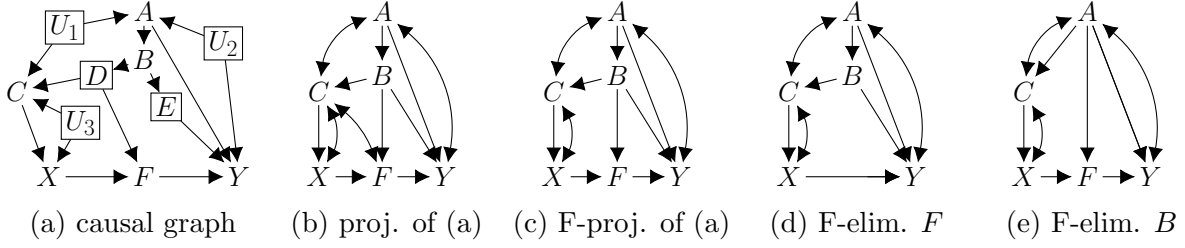


Figure 3.4: Variables A, B, C, F, X, Y are observed. Variables D, E are functional.

so that identifiability becomes unconstrained by functional dependencies. If we reach this point, we can use existing algorithms for classical identifiability, like the ID algorithm, to test identifiability.⁵ This can be subtle though since hidden functional variables behave differently from observed ones. We start with the case of eliminating hidden variables.

Theorem 2. *Let $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ be an identifiability tuple. If G' is the result of functionally eliminating the hidden functional variables ($\mathbf{W} \setminus \mathbf{V}$) from G , then the causal effect of $\mathbf{X}$ on $\mathbf{Y}$ is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ iff it is identifiable wrt $\langle G', \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{V} \cap \mathbf{W}\} \rangle$.*

That is, we can always eliminate all hidden functional variables while preserving (constrained) identifiability. An immediate corollary of this theorem is that if all functional variables are hidden, then we can reduce the problem of constrained identifiability (under functional dependencies) to classical identifiability since $\mathbf{V} \cap \mathbf{W} = \emptyset$.

Corollary 3. *Let $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ be an identifiability tuple where $\mathcal{C}_{\mathbf{V}} = \{\Pr(\mathbf{V}) > 0\}$ and $\mathbf{W}$ are all hidden. If G' is the result of functionally projecting G on $\mathbf{V}$, then the causal effect of $\mathbf{X}$ on $\mathbf{Y}$ is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ iff it is identifiable wrt $\langle G, \mathbf{V}, \mathcal{C}_{\mathbf{V}} \rangle$.⁶*

This corollary suggests a method for using the ID algorithm, which is popular for testing classical identifiability, to establish identifiability with functional dependencies by coupling ID with functional projection instead of classical projection. Consider the causal graph G

⁵In this chapter, *classical identifiability* refers to Definition 1 with positivity constraints (e.g., $\Pr(\mathbf{V}) > 0$).

⁶We are requiring the positivity constraint $\Pr(\mathbf{V}) > 0$ as we suspect the projection operation in [TP02b] requires it (even though that was not made explicit in the paper). If not, then $\mathcal{C}_{\mathbf{V}}$ can be empty in Corollary 3.

in Figure 3.4(a) with observed variables $\mathbf{V} = \{A, B, C, F, X, Y\}$. The causal effect of X on Y is not identifiable under $\Pr(\mathbf{V}) > 0$: projecting G on observed variables $\mathbf{V}$ yields the causal graph G' in Figure 3.4(b) and the ID algorithm produces FAIL on G' . Suppose now that the hidden variables $\{D, E\}$ are functional. To test the identifiability of causal effect using Corollary 3, we functionally project G on the observed variables $\mathbf{V}$ which yields the causal graph G'' in Figure 3.4(c). Applying the ID algorithm to G'' produces the following identifying formula: $\Pr_x(y) = \sum_{bf} \Pr(f|b, x) \sum_{acx'} \Pr(y|a, b, c, f, x') \Pr(a, b, c, x')$ so $\Pr_x(y)$ is identifiable under the functional dependencies.

3.5 Treating Observed Functional Variables

We now treat the case when some of the functional variables are observed. The subtlety here is that, unlike hidden functional variables, eliminating an observed functional variable does not always preserve identifiability. However, the following result identifies conditions that guarantees the preservation of identifiability based on the notion of separability in Definition 5. If all observed functional variables satisfy these conditions, we can again reduce identifiability with functional dependencies into classical identifiability so we can exploit existing methods for identifiability like the ID and IDENTIFY algorithms.

Theorem 3. *Let $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ be an identifiability tuple. Let $\mathbf{Z}$ be a set of observed functional variables that are neither treatments nor outcomes, are separable from $\mathcal{C}_{\mathbf{V}}$, and that have observed parents. If G' is the result of functionally eliminating variables $\mathbf{Z}$ from G , then the causal effect of $\mathbf{X}$ on $\mathbf{Y}$ is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ iff it is identifiable wrt $\langle G', \mathbf{V} \setminus \mathbf{Z}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W} \setminus \mathbf{Z}\} \rangle$.*

Intuitively, the theorem allows us to remove observed functional variables from a causal graph if they satisfy the given conditions. If these conditions are violated, for example when a variable with a hidden parent is eliminated, identifiability may not be preserved; we formalize this result as Proposition 13 in Appendix B.2. We now have the following

important corollary of Theorems 2 and 3, which subsumes Corollary 3.

Corollary 4. *Let $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ be an identifiability tuple where $\mathcal{C}_{\mathbf{V}} = \{\Pr(\mathbf{V} \setminus \mathbf{W}) > 0\}$ and every variable in $\mathbf{W} \cap \mathbf{V}$ satisfies the conditions of Theorem 3. If G' is the result of functionally projecting G on $\mathbf{V} \setminus \mathbf{W}$, then the causal effect of $\mathbf{X}$ on $\mathbf{Y}$ is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ iff it is identifiable wrt $\langle G', \mathbf{V} \setminus \mathbf{W}, \mathcal{C}_{\mathbf{V}} \rangle$.*

Consider again the causal effect of X on Y in graph G of Figure 3.4(a) with observed variables $\mathbf{V} = \{A, B, C, F, X, Y\}$. Suppose now that the observed variable F is also functional (in addition to the hidden functional variables D, E) and assume $\Pr(A, B, C, X, Y) > 0$. Using Corollary 4, we can functionally project G onto $\{A, B, C, X, Y\}$ to yield the causal graph G' in Figure 3.4(d), which reduces identifiability on G to classical identifiability on G' . Since strict positivity holds in G' , we can apply any existing identifiability algorithm and conclude that the causal effect is not identifiable. For another scenario, suppose that the observed variable B (instead of F) is functional and we have $\Pr(A, C, F, X, Y) > 0$. Again, using Corollary 4, we functionally project G onto $\{A, C, F, X, Y\}$ to yield the causal graph G'' in Figure 3.4(e), which reduces identifiability on G to classical identifiability on G'' . If we apply the ID algorithm to G'' we get the identifying formula (which we denote as Eq. 1): $\Pr_x(y) = \sum_{af} \Pr(f|a, x) \sum_{cx'} \Pr(y|a, c, f, x') \Pr(a, c, x')$. In both scenarios above, we were able to test identifiability under functional dependencies using existing algorithms.

Corollary 4 (and Theorem 3) has yet another key application: it can help us pinpoint observations that are not essential for identifiability. To illustrate, consider the second scenario above where the observed variable B is functional in the causal graph G of Figure 3.4(a). The fact that Corollary 4 allowed us to eliminate variable B from G implies that observing this variable is not needed for rendering the causal effect identifiable and, hence, is not needed for computing the causal effect. This can be seen by examining the identifying formula (Eq. 1) which does not

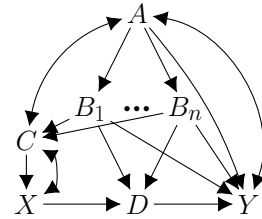


Figure 3.5: $B_1, \dots, B_n$ are functional.

contain variable B . This can be further generalized to the causal graph in Figure 3.5 with functional variables $B_1, \dots, B_n$ and assume $\Pr(A, C, X, D, Y) > 0$. By Corollary 4, we can functionally project the graph onto $\mathbf{V}' = \{A, C, X, D, Y\}$ while preserving the identifiability. Moreover, applying the ID algorithm to G yields an identifying formula for $\Pr_x(Y)$ over only $\mathbf{V}'$. That is, in this example, we only need to observe a constant number (five) of variables to render the causal effect identifiable given the functional dependencies, even though the number of observed variables in the original graph is unbounded. This application of Corollary 4 can be quite significant in practice, especially when some variables are expensive to measure (observe), or when they may raise privacy concerns; see, e.g., [TK17a, ZLT19].

Theorems 2 and 3 are more far-reaching than what the above discussion may suggest. In particular, even if we cannot eliminate every (observed) functional variable using these theorems, we may still be able to identify the causal effects due to the following result.

Theorem 4. *Let $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ be an identifiability tuple. If every functional variable has at least one hidden parent, then a causal effect of $\mathbf{X}$ on $\mathbf{Y}$ is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ iff it is identifiable wrt $\langle G, \mathbf{V}, \mathcal{C}_{\mathbf{V}} \rangle$.*

That is, if we still have functional variables in the causal graph after applying Theorems 2 and 3, and if each such variable has at least one hidden parent, we can ignore the functional dependencies on these variables. Consider again the causal effect of X on Y in G of Figure 3.4(a) with observed variables $\mathbf{V} = \{A, B, C, F, X, Y\}$. Suppose now that the observed variables A, B, C, X, Y are also functional (in addition to the hidden functional variables D, E) and assume $\Pr(A, C, F, X, Y) > 0$. By Theorem 3, we first reduce identifiability on G to that on G' in Figure 3.4(e) by functionally eliminating D, E, B . Since all the remaining functional variables A, C, X, Y have some hidden parent, we can further reduce it to classical identifiability on G' by Theorem 4 and then apply existing algorithms to conclude that the causal effect is identifiable.

3.6 A Different Approach Under Weaker Positivity Assumptions

The method we presented thus far for testing identifiability under functional dependencies is based on eliminating functional variables from the causal graph, followed by applying existing tools for causal effect identification such as the project-ID algorithm and the do-calculus. This identifiability method is complete if every observed functional variable either satisfies the conditions of Theorem 3 or has at least one hidden parent that is not functional.

We next present a different approach for eliminating functional dependencies from the identifiability problem. This method is more direct than the previous one, but it assumes a weaker positivity constraint and hence does not allow us to fully exploit some existing tools like the ID algorithm. Intuitively, the new method is based on pretending that some of the hidden functional variables are actually observed and is inspired by Proposition 2 which reduces D-separation to d-separation using a similar technique.

Theorem 5. *Let $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ be an identifiability tuple where $\mathcal{C}_{\mathbf{V}} = \{\Pr(X) > 0, X \in \mathbf{X}\}$. A causal effect of $\mathbf{X}$ on $\mathbf{Y}$ is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ iff it is identifiable wrt $\langle G, \mathbf{V}', \mathcal{C}_{\mathbf{V}} \rangle$ where $\mathbf{V}'$ is obtained as follows. Initially, $\mathbf{V}' = \mathbf{V}$. Repeat until $\mathbf{V}'$ stops changing: add to $\mathbf{V}'$ a functional variable from $\mathbf{W}$ if all of its parents are in $\mathbf{V}'$.*

Consider the causal effect of X on Y in graph G of Figure 3.4(a) and suppose the observed variables are $\mathbf{V} = \{A, B, C, X, Y\}$, the functional variables are $\{D, E, F\}$ and we have a weaker positivity constraint $\Pr(X) > 0$. By Theorem 5, the causal effect of X on Y is identifiable with functional dependencies iff it is identifiable without functional dependencies in G while pretending that variables $\mathbf{V}' = \{A, B, C, D, E, F, X, Y\}$ are all observed. In this case, the causal effect is not identifiable but we cannot obtain this answer by applying an identifiability algorithm that requires positivity constraints that are stronger than $\Pr(X) > 0$. If we have stronger positivity constraints that imply $\Pr(X) > 0, X \in \mathbf{X}$, then only the if part of Theorem 5 will hold, assuming $\mathcal{C}_{\mathbf{V}}$ and $\mathbf{W}$ are consistent. That is, confirming identifiability wrt $\langle G, \mathbf{V}', \mathcal{C}_{\mathbf{V}} \rangle$ guarantees identifiability wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$;

however, the latter may still hold even if the former does not. This suggests that, to fully exploit the power of Theorem 5, one would need a new class of identifiability algorithms that can operate under the weakest possible positivity constraints.

3.7 Experiments

We report on a simple experiment to empirically demonstrate how knowledge of functional dependencies can aid the identifiability of causal effects. We randomly generated 50 causal graphs (DAGs) with $N \in \{50, 100, 150\}$ variables using the Erdos-Renyi method [ER59], where every edge in the causal graphs appears with a probability of 0.1 and every variable has at most 6 parents. We then randomly picked $0.8N$ observed variables, $0.2N$ treatments, $0.2N$ outcomes, and $W \in \{0.25N, 0.5N, 0.75N\}$ functional variables from the causal graphs.

For each combination of N and W , Table 3.2 records the number of causal effects (out of 50) that are (1) unidentifiable (UID); and (2) unidentifiable without functional dependencies but identifiable with functional dependencies (UID-FID). The table also records the average number of observed variables after applying Theorems 2–4 ($\#OBS$); these are the observed variables passed to the project-ID algorithm.⁷

The following patterns are clear. First, more unidentifiable causal effects become identifiable when more variables exhibit functional dependencies. This observation demonstrates that knowledge of functional dependencies can greatly improve the identifiability of causal effects. Secondly, the number of observed variables required by the project-ID algorithm becomes smaller when there are more functional variables, implying that we only need to collect data on a smaller set of variables to estimate the (identifiable) causal effects. Again, this is because more observed functional variables can be functionally eliminated from the causal graphs by Theorem 3.

⁷We assume that strict positivity holds for the remaining observed variables after applying the theorems.

Table 3.2: Numbers of causal effects that are unidentifiable (UID) and that become identifiable under functional dependencies (UID-FID), and the average number of observed variables passed to project-ID ($\#OBS$), for causal graphs with varying numbers of variables (N) and functional variables (W).

N	W	UID	UID-FID	$\#OBS$
50	$0.25N$	15	3	36.5
	$0.5N$	15	10	32.6
	$0.75N$	15	11	27.3
100	$0.25N$	50	0	74.7
	$0.5N$	50	6	68.1
	$0.75N$	50	18	57.3
150	$0.25N$	50	0	113.2
	$0.5N$	50	2	103.6
	$0.75N$	50	5	89.7

CHAPTER 4

Identifying Causal Effects Under Context-Specific Independencies

Context-specific independencies (CSIs) are another type of knowledge that specify independencies within parameters. These independencies cannot be captured by causal graphs and can therefore be formulated as additional constraints on the models. We first show in Section 4.1 that under CSIs, it is meaningful to consider a more granular type of identifiability, which examines how an intervention on a particular *state* of treatments affects a particular *state* of outcomes. This is in contrast to classical causal effects in Definition 1, where treatments and outcomes are defined as variables. We illustrate with examples in Section 4.2 that the state-based causal effects may be identifiable even when the classical variable-based causal effects are not. We then present another type of knowledge that constrains the possible states of variables and demonstrate in Section 4.3 its role in improving both levels of identifiability when combined with CSIs. We propose an approach for identifying causal effects under these constraints in Section 4.4 and conduct empirical studies to further illustrate the separation between the two levels of identifiability in Section 4.5. The results in this chapter appeared in [CD25b].

4.1 Two Levels of Causal Effect Identifiability

The definition of classical identifiability (Definition 1) is variable-based in the following sense. If the causal effect $\Pr_{\mathbf{X}}(\mathbf{Y})$ is identifiable, then $\Pr_{\mathbf{X}}(\mathbf{y})$ is uniquely computable from the

observational distribution $\Pr(\mathbf{V})$ for *every* possible treatment $\mathbf{x}$ and *every* possible outcome $\mathbf{y}$. The constrained identifiability in Definition 3 imposes restrictions to the models considered for identifiability but remains variable-based.

We now introduce a more granular notion of state-based identifiability which targets a particular treatment $\mathbf{x}$ and a particular outcome $\mathbf{y}$. For example, the probability that a major accident will occur if a particular speed limit is enforced, $\Pr_{\text{speed}=65}(\text{accident}=\text{major})$. We will adapt the definition of constrained identifiability here, as classical identifiability can be treated as a special case when the constraint set $\mathcal{C}$ is empty.

Definition 8 (State-Based Identifiability). *Let $\mathcal{C}$ be a set of constraints. The causal effect of $\mathbf{x}$ on $\mathbf{y}$, denoted $\Pr_{\mathbf{x}}(\mathbf{y})$, is (constrained) identifiable wrt $\langle G, \mathbf{V}, \mathcal{C} \rangle$ if $\Pr_{\mathbf{x}}^1(\mathbf{y}) = \Pr_{\mathbf{x}}^2(\mathbf{y})$ for any pair of models that satisfy $\mathcal{C}$ and induce distributions $\Pr^1, \Pr^2$ where $\Pr^1(\mathbf{V}) = \Pr^2(\mathbf{V})$.*

Throughout this chapter, we always assume the strict positivity constraint $\Pr(\mathbf{V}) > 0$ and will not list it explicitly as a constraint in $\mathcal{C}$. State-based identifiability refers to a particular treatment $\mathbf{x}$ and a particular outcome $\mathbf{y}$ which are states of the treatment and outcome variables. The distinction between variable-based and state-based identifiability is not interesting if the only information available consists of a causal graph G and an observational distribution $\Pr(\mathbf{V})$. In fact, the two notions are equivalent in this case, which we formulate as the following proposition.

Proposition 3. *Consider the definition of constrained identifiability and suppose the set of constraints $\mathcal{C}$ is empty. Then the causal effect $\Pr_{\mathbf{x}}(\mathbf{y})$ is (variable-based) identifiable iff $\Pr_{\mathbf{x}}(\mathbf{y})$ is (state-based) identifiable for some states $\mathbf{x}, \mathbf{y}$.*

The distinction between the two notions materializes though if we have additional knowledge beyond G and $\Pr(\mathbf{V})$, as we elaborate next.

4.2 The Impact of CSI Constraints on Identifiability

We show how CSI constraints [BFG96, SCD20], studied for causal effect identifiability in [THK19, MJE22, CD25a], draw a distinction between variable-based and state-based identifiability. We consider CSIs in the form of $(Y \perp\!\!\!\perp P_1 \mid \mathbf{p}_2, \mathbf{P}_3)$, where Y is a variable in the causal graph and $\{P_1\}, \mathbf{P}_2, \mathbf{P}_3$ form a partition of its parents. This CSI constraint is interpreted as follows: Y is independent of P_1 under the specific state $\mathbf{p}_2$ of $\mathbf{P}_2$ and any states of $\mathbf{P}_3$. We call $\mathbf{P}_2$ the *context variables* and $\mathbf{p}_2$ the *context* of the CSI. For example, the CSI $(\text{Light} \perp\!\!\!\perp \text{People} \mid \text{Outage}=\text{yes}, \text{Time})$ states that when a power outage occurs (context), the status of light becomes independent of the presence of people, regardless of the time of day. This independence, however, does not hold when the power outage does not occur. Another example is $(\text{Heater} \perp\!\!\!\perp \text{Occupants} \mid \text{Temperature}=\text{high})$, which states that the heater status becomes independent of the number of occupants in the room when the outdoor temperature is high. CSI constraints are effectively constraints on model parameters so they cannot be captured using the language of causal graphs. Constraints, such as CSIs, restrict the set of models considered when deciding identifiability. This has the impact of making some causal effects identifiable, which would be unidentifiable without the constraints.

Our first result states that the state-based identifiability and the variable-based identifiability are equivalent if none of the states $\mathbf{x}, \mathbf{y}$ appear in the contexts of CSIs.

Proposition 4. *Consider the definition of constrained identifiability and suppose $\mathcal{C}$ is a set of CSI constraints. If none of the states $\mathbf{x}, \mathbf{y}$ appear in the contexts of $\mathcal{C}$, then $\text{Pr}_{\mathbf{x}}(\mathbf{y})$ is state-based identifiable wrt $\langle G, \mathbf{V}, \mathcal{C} \rangle$ iff $\text{Pr}_{\mathbf{x}}(\mathbf{Y})$ is variable-based identifiable wrt $\langle G, \mathbf{V}, \mathcal{C} \rangle$.*

For example, if $(\text{Light} \perp\!\!\!\perp \text{People} \mid \text{Outage}=\text{yes}, \text{Time})$ is the only CSI constraint, then $\text{Pr}_{\text{Outage}=\text{no}}(\text{Light}=\text{off})$ is identifiable iff $\text{Pr}_{\text{Outage}}(\text{Light})$ is identifiable.

It immediately follows that the (variable-based) identifiability of $\text{Pr}_{\mathbf{x}}(\mathbf{Y})$ can be reduced to the (state-based) identifiability of $\text{Pr}_{\mathbf{x}}(\mathbf{y})$ for states $\mathbf{x}, \mathbf{y}$ that do not appear in CSI contexts. Hence, if we have an algorithm for testing state-based identifiability under CSIs, we can use

it to test variable-based identifiability under CSIs by choosing states $\mathbf{x}, \mathbf{y}$ that do not appear in CSI contexts. This can always be done as we are only considering CSI constraints and these do not constrain variable states.

However, in general, state-based and variable-based identifiability are not equivalent. We demonstrate the separation between the two using the following example. Consider the causal graph in Figure 4.1(a) where **Age** and **Degree** are hidden, and **Years** refers to years of experience. Suppose now that the company passes a policy that ignores the employee’s degree when they decide the salary for entry-level employees, which gives the CSI $(S \perp \perp D \mid Y, J=\text{entry-level})$. The state-based causal effect $\Pr_{Y \geq 10}(J=\text{entry-level}, S=\text{low})$ is now identifiable; see Proposition 16 in Appendix C.3. However, a different state-based causal effect $\Pr_{Y \geq 10}(J=\text{senior-level}, S=\text{low})$ is unidentifiable since we cannot leverage the CSI when $J=\text{senior-level}$ (the CSI only takes effect when $J=\text{entry-level}$). Therefore, the variable-based causal effect $\Pr_Y(J, S)$ is unidentifiable yet the state-based causal effect $\Pr_{Y \geq 10}(J=\text{entry-level}, S=\text{low})$ is identifiable.

This example shows that an unidentifiable causal effect at the variable level may be identifiable for a specific treatment and outcome. Hence, existing methods for variable-based identifiability are no longer sufficient for state-based identifiability, highlighting the need for more refined approaches to identify causal effects at this more granular level.

The next question we consider is whether CSI constraints can improve variable-based identifiability like they do for state-based identifiability. That is, if a causal effect is unidentifiable (according to Definition 1), can it become identifiable when CSI constraints are imposed (according to Definition 3)? The answer turns out to be positive. Consider the modified causal graph for the company example depicted in Figure 4.1(b), where Y is no longer a parent of S . In this graph, the variable-based causal effect $\Pr_J(S)$ is unidentifiable. However, suppose we further know that the company determines an employee’s job level only based on their age if they have worked for over 10 years — i.e., we have the CSI $(J \perp \perp D \mid A, Y \geq 10)$. The same causal effect becomes identifiable and can be estimated as

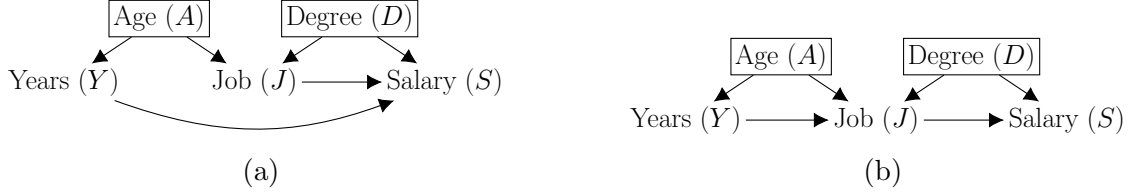


Figure 4.1: Causal graphs showing the effectiveness of CSI constraints, with hidden variables shown in rectangles.

$\Pr_J(S) = \Pr(S|J, Y \geq 10)$. We formalize this result as Proposition 17 in Appendix C.3. The result suggests that classical identification methods (such as **IDENTIFY** [TP03]) are insufficient for variable-based identifiability when additional CSIs are available.

4.3 The Impact of State Constraints on Identifiability

We showed earlier that CSI constraints alone can improve both variable-based and state-based identifiability. We next show that these constraints can further improve identifiability when coupled with *state constraints*, which specify the possible states of some variables. Note that CSI constraints do not in principle require the specification of variable states (i.e., set of all states) even for variables that participate as contexts in such constraints. For example, the CSI (**Light** $\perp$ **People** | **Outage** = *yes*) only requires “yes” to be a valid state of the variable **Outage** but does not require specifying the other states of **Outage**. Note also that earlier works in [THK19, MJE22], which studied the impact of CSI constraints on causal effect identifiability, did assume that all variables are binary which is effectively a state constraint. Proposition 18 in Appendix C.3 shows that state constraints have no impact on variable-based or state-based identifiability without additional knowledge (e.g., CSIs), assuming the causal graph is semi-Markovian.¹ This perhaps explains why they did not receive as much attention in the literature on causal effect identifiability. We next consider

¹In a Semi-Markovian graph, every hidden variable is a root and has exactly two children [Ver93, TP02b]. This assumption is quite common in the existing literature, e.g., the **IDENTIFY** algorithm in [TP03, HV06] and the **ID** algorithm in [SP06].

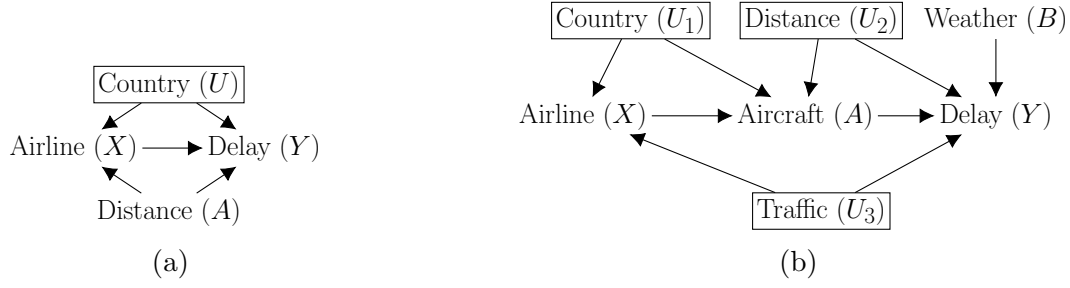


Figure 4.2: Causal graphs showing the effectiveness of state constraints when coupled with CSIs.

the interplay between CSIs and state constraints though, showing that the latter can indeed improve both variable-based and state-based identifiability when combined with the former.

Consider the causal graph in Figure 4.2(a) which models flight delays. Here, **Country** (U) refers to the country of departure and is assumed to be hidden. Assume further the CSI constraints $(X \perp\!\!\!\perp U \mid A=short)$ and $(Y \perp\!\!\!\perp U \mid X, A=long)$. Without restricting variable states, the variable-based causal effect $\Pr_X(Y)$ is not identifiable. Moreover, no state-based causal effect $\Pr_x(y)$ is identifiable in this case. However, if we assume the states of A to be $\{short, long\}$, the causal effect $\Pr_X(Y)$ becomes identifiable and, hence, also state-based identifiable for any states x, y . In fact, we can estimate the causal effect using the following formula: $\Pr_X(Y) = \Pr(A=short) \Pr(Y|X, A=short) + \Pr(A=long)\Pr(Y|X, A=long)$. This example is formalized as Proposition 19 in Appendix C.3. Interestingly, the state constraint of this example involves an observed variable yet it still managed to improve identifiability when coupled with CSIs.

The next example shows that adding state constraints to CSIs may enable state-based identifiability but not variable-based identifiability. Consider the causal graph in Figure 4.2(b) which depicts a more detailed model of flight delays. Suppose U_1, U_2, U_3 are hidden and we have the following CSIs: $(A \perp\!\!\!\perp U_1 \mid X=low-cost, U_2)$, $(Y \perp\!\!\!\perp U_3 \mid B=rainy, U_2, A)$, and $(Y \perp\!\!\!\perp A \mid B=sunny, U_2, U_3)$. Without state constraints, the causal effect $\Pr_x(y)$ is unidentifiable for any states x, y (hence, $\Pr_X(Y)$ is not identifiable). If we assume that **Weather** (B) has the only states $\{rainy, sunny\}$, then the causal effect becomes state-based identifiable but still

not variable-based identifiable. In particular, the state constraint enables the identifiability of $\Pr_{X=low-cost}(Y=true)$ but not $\Pr_{X=high-cost}(Y=true)$ so $\Pr_X(Y)$ remains unidentifiable. This is formalized as Proposition 20 in Appendix C.3.

4.4 Testing Identifiability Under CSI and State Constraints

We have so far discussed the role of additional constraints – CSIs and state constraints – in improving both variable-based and state-based identifiability. More importantly, we have demonstrated the separation between these two levels of identifiability, both under CSIs alone and when the two types of constraints are combined. We next introduce a systematic method for testing state-based identifiability in the presence of CSI and state constraints. We start by presenting an algorithm for testing identifiability under CSI constraints only, and then extend it in Section 4.4.2 to treat state constraints as well. For the case of CSI constraints only, we immediately obtain a method for testing variable-based identifiability by considering specific states of treatments and outcomes based on Proposition 4. For the case with additional state constraints, we can similarly test variable-based identifiability using the method for state-based identifiability as we demonstrate in Section 4.4.2. One assumption made by our method is that all contexts appearing in the CSIs are observed. This assumption commonly holds when the states of hidden variables are completely unknown (i.e., unconstrained) and therefore cannot appear in the CSI contexts.²

4.4.1 Leveraging CSI Constraints

Algorithm 1 shows the procedure for testing state-based identifiability, assuming the only constraints are CSIs. The main procedure ID-CSI takes as input the standard components for classical identifiability (a causal graph, observed variables, and treatment and outcome

²This assumption is stronger than [THK19] which allows hidden contexts, and is weaker than [MJE22] which assumes all contexts are observed roots in the graph.

states), along with a set of CSI constraints. The algorithm returns a formula for estimating the causal effect $\Pr_{\mathbf{x}}(\mathbf{y})$ if it is identifiable; otherwise, it returns FAIL. The structure of the algorithm is similar to that of IDENTIFY [TP03]: a causal graph is decomposed into sub-graphs, called *c-components*, and the problem of causal effect identification is converted to independent identification problems on these c-components. However, there are two key differences between Algorithm 1 and IDENTIFY. First, Algorithm 1 computes the c-components of a novel notion called *context-induced graph*, which is a pruned version of the causal graph based on CSIs. Second, the algorithm utilizes a modified procedure for the identification problem on c-components (of context-induced graphs) which is specified by the ID-QFUNC subroutine. We elaborate on both differences below. A detailed review of c-components and the associated notion of c-factors is provided in Appendix A.2.

Identifying Causal Effects Using Context-Induced Graphs

We start with the notion of context-induced graphs (utilized by Algorithm 1), which simplify the causal graph by leveraging CSI constraints. The definition of context-induced graph (CI-graph) takes as input a set of CSIs and a context and removes edges based on the CSIs activated by the context.

Definition 9. *Let G be a causal graph, $\mathcal{C}$ be a set of CSIs, and $\mathbf{s}$ be a context. A CI-graph under $\mathbf{s}$ is obtained from G by repeatedly removing edges as follows: remove edge $P_1 \rightarrow C$ if there exists a CSI $(C \perp\!\!\!\perp P_1 \mid \mathbf{p}_2, \mathbf{P}_3)$ for which $\mathbf{p}_2 \subseteq \mathbf{s}$ and $\mathbf{P}_2$ are parents of C at the time of removal.*

Given a fixed context, the choice of CI-graph may not be unique. To illustrate, consider the causal graph in Figure 4.3(a) and assume the CSIs $(D \perp\!\!\!\perp B \mid c_0, A)$, $(D \perp\!\!\!\perp C \mid b_0, A)$, and $(E \perp\!\!\!\perp B \mid c_0)$. The graphs in Figures 4.3(b) and 4.3(c) are both valid CI-graphs under the context $b_0 c_0$. In particular, Figure 4.3(b) is obtained by first removing the edge $B \rightarrow E$ based on the CSI $(E \perp\!\!\!\perp B \mid c_0)$, and then removing the edge $B \rightarrow D$ based on $(D \perp\!\!\!\perp B \mid c_0, A)$.

Algorithm 1 State-Based Identifiability with CSI Constraints

```

1: procedure ID-CSI( $G, \mathbf{V}, \mathbf{x}, \mathbf{y}, \mathcal{C}$ )
  // input: causal graph  $G$ , observed variables  $\mathbf{V}$ , treatment  $\mathbf{x}$ , outcome  $\mathbf{y}$ , CSI constraints  $\mathcal{C}$ 
  // output: identifying formula for  $\Pr_{\mathbf{x}}(\mathbf{y})$  or FAIL
  // Assume  $\mathbf{V} = (V^1, V^2, \dots)$  are ordered topologically in  $G$ , and let  $\mathbf{V}^{(i)} = \{V^1, \dots, V^i\}$ 
2:   for each CI-graph  $T$  under context  $\mathbf{xy}$  do
3:      $M \leftarrow$  remove  $\mathbf{X}$  and retain the subgraph of  $T$  consisting of all ancestors of  $\mathbf{Y}$ 
4:      $\mathbf{C}_1, \dots, \mathbf{C}_m \leftarrow$  maximal c-components in  $M$ 
5:     for  $i = 1$  to  $m$  do
6:       for each context  $\mathbf{s}$  and its CI-graph  $H$  do
7:         if  $\mathbf{C}_i$  have the same parents in  $T$  and  $H$  and  $(\mathbf{s} \setminus \mathbf{xy})$  contains no state of  $\mathbf{C}_i$  or their
           parents then
8:            $f^i \leftarrow$  ID-QFUNC( $\mathbf{C}_i, \mathbf{V} \cup \mathbf{U}, \Pr(\mathbf{V}), \mathbf{s}, H$ )  //  $\mathbf{U}$  are the hidden variables
9:           if  $f^i \neq \text{FAIL}$  then break
10:        if  $f^i \neq \text{FAIL}$  for  $i = 1, \dots, m$  then
11:          return  $\sum_{\mathbf{V} \setminus (\mathbf{x} \cup \mathbf{y})} \prod_{i=1}^m f_{\mathbf{xy}}^i$ 
12:    return FAIL

13: procedure ID-QFUNC( $\mathbf{C}, \mathbf{D}, f, \mathbf{s}, H$ )
  // input: c-component  $\mathbf{C}$ , variable set  $\mathbf{D}$ , expression (factor)  $f$ , context  $\mathbf{s}$ , graph  $H$ 
  // output: formula for computing the c-factor  $f^i$  or FAIL
14:   if  $\mathbf{D} = \mathbf{C}$  then return  $f_{\mathbf{s}}$ 
15:    $H_{\mathbf{D}} \leftarrow$  subgraph of  $H$  formed by variables in  $\mathbf{D}$ 
16:   if exists a leaf node  $W$  in  $H_{\mathbf{D}}$  and  $W \notin \mathbf{S} \cup \mathbf{C}$  then
17:      $\mathbf{D}' \leftarrow \mathbf{D} \setminus \{W\}$ 
18:      $f' \leftarrow \sum_W f$ 
19:     return ID-QFUNC( $\mathbf{C}, \mathbf{D}', f', \mathbf{s}, H$ )
20:   if exists a maximal c-component  $\mathbf{D}'$  in  $H_{\mathbf{D}}$  such that  $\mathbf{C} \subseteq \mathbf{D}' \subset \mathbf{D}$  then
21:      $f' \leftarrow \prod_{V^i \in \mathbf{D}' \cap \mathbf{V}} \frac{\sum_{\mathbf{D} \setminus \mathbf{V}^{(i)}} f}{\sum_{\mathbf{D} \setminus \mathbf{V}^{(i-1)}} f}$ 
22:     return ID-QFUNC( $\mathbf{C}, \mathbf{D}', f', \mathbf{s}, H$ )
23:   return FAIL
  
```

On the other hand, Figures 4.3(b) applies the same first step but removes the edge $C \rightarrow D$ based on $(D \perp\!\!\!\perp C \mid b_0, A)$ as the second step. No more edges can be removed from either graph. CI-graphs reduce the parameters of a causal graph by removing some of its edges, while preserving the (conditional) independencies implied by the original causal graph and CSI constraints given the context; see Appendix C.1 for further details.

Instead of testing identifiability using the original causal graph, Algorithm 1 enumerates all possible CI-graphs T under the context $\mathbf{xy}$ and checks whether the causal effect can be identified using any of them (line 2). Similar to the classical identification method

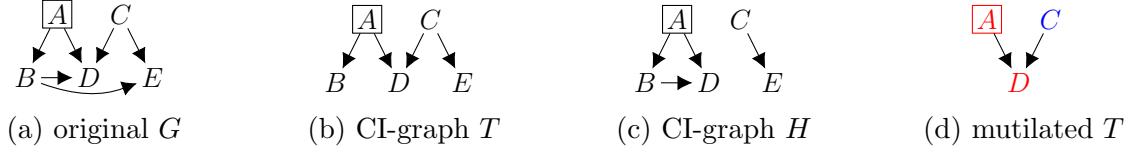


Figure 4.3: A causal graph, distinct context-induced graphs, and a mutilated graph under the context b_0c_0 .

IDENTIFY [TP03], the CI-graph T is then mutilated by removing the treatments $\mathbf{X}$ and retaining the subgraph consisting of the ancestors of outcomes $\mathbf{Y}$ (line 3). If we decompose the resulting graph into c-components $\{\mathbf{C}_1, \dots, \mathbf{C}_m\}$, we obtain the following key property: $\Pr_{\mathbf{x}}(\mathbf{y}) = \sum_{\mathbf{V} \setminus (\mathbf{X} \cup \mathbf{Y})} \prod_{i=1}^m f_{\mathbf{x}\mathbf{y}}^i$, where $f_{\mathbf{x}\mathbf{y}}^i$ denotes the entries of the c-factor for $\mathbf{C}_i$ that are compatible with $\mathbf{x}\mathbf{y}$.³ Hence, if all c-factors corresponding to the c-components in the (mutilated) CI-graph are identifiable from the observational distribution $\Pr(\mathbf{V})$, the causal effect is also identifiable. This yields a sound method for testing identifiability by checking identifiability with respect to each independent c-component in the CI-graph. We state this result formally as Lemma 5 with proof in Appendix C.3.

To illustrate, consider the causal graph in Figure 4.1(a) with the CSI $(S \perp\!\!\!\perp D \mid Y, j_0)$ and state-based causal effect $\Pr_{y_0}(j_0, s_0)$. Figures 4.4(a) and 4.4(b) depict a CI-graph under the context $y_0j_0s_0$ (treatment and outcome states) and its mutilated graph, where $\{A, D, J\}$ and $\{S\}$ are the c-components in the mutilated graph (highlighted with colors). Based on our discussion earlier, the causal effect is identifiable if the corresponding c-factors are identifiable. Consider now a different example in Figure 4.1(b) with CSI $(S \perp\!\!\!\perp D \mid Y, j_0)$ and causal effect $\Pr_{y_0}(j_0, s_0)$. Figures 4.4(c) and 4.4(d) depict the CI-graph and its mutilated graph under the context $y_0j_0s_0$. Since $\{D, S\}$ is the only c-component in the mutilated graph, the causal effect is identifiable if the c-factor is identifiable. One key observation is that, since CI-graphs are always subgraphs of the original graph, the c-components (and corresponding

³We assume that the c-components contain both observed and hidden variables. The c-factor for each c-component $\mathbf{C}_i$ in graph T is defined as $f^i = \sum_{\mathbf{U}_i} \prod_{C \in \mathbf{C}_i} \theta_{C|\mathbf{P}_C}$, where $\mathbf{U}_i$ denotes the hidden variables in $\mathbf{C}_i$ and $\mathbf{P}_C$ denotes the parents of C in graph T . For example, in Figure 4.4(a), the c-factor for the c-component $\{A, D, J\}$ can be computed as $f = \sum_{AD} \theta_A \theta_D \theta_{J|AD}$, and its entries under the context $y_0j_0s_0$ are given by $f_{y_0j_0s_0} = \sum_{AD} \theta_A \theta_D \theta_{j_0|AD}$. See Appendix A.2 for additional examples.

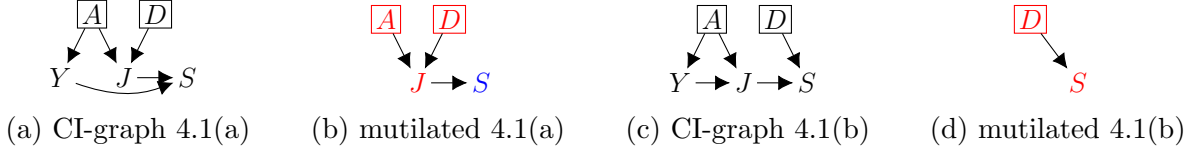


Figure 4.4: Context-induced and mutilated graphs for Figure 4.1.

c-factors) extracted from the mutilated CI-graphs are no larger than those in the original graph, which intuitively allows more causal effects to be identified.

Treating C-Components of CI-Graphs

We have shown that the problem of identifying causal effects can be converted to that of identifying c-factors induced by the c-components in the CI-graph. We next present a method for testing the identifiability of these c-factors, which involves the ID-QFUNC procedure in Algorithm 1.

A graph-based method for testing c-factor identifiability was introduced in IDENTIFY [TP03]. This method applies to the c-factors obtained from the original causal graphs but does not extend to c-factors corresponding to the c-components of CI-graphs. We illustrate this limitation with a concrete example in Appendix C.3 Proposition 15. Here, we propose a novel method for treating the identification of c-components (and their corresponding c-factors) within CI-graphs by modifying the approach in IDENTIFY. As we demonstrate later, our method is sound for c-factor identification, and is complete for classical identifiability (without constraints) when combined with the procedure described earlier.

Our goal is to check the identifiability of the c-factor for each c-component $\mathbf{C}_i$ in the (mutilated) CI-graph T . To achieve this, we start by enumerating all possible contexts $\mathbf{s}$ and their CI-graphs H (line 6). We then check whether (i) variables in $\mathbf{C}_i$ have the same parents in T and H , and (ii) $(\mathbf{s} \setminus \mathbf{xy})$ contains no state of $\mathbf{C}_i$ or of their parents (line 7). The intuition here is that if both conditions are satisfied, the c-factor induced by $\mathbf{C}_i$ in H (denoted $f_{\mathbf{s}}^i$)

must contain all the entries of the c-factor of interest (denoted $f_{\mathbf{xy}}^i$).⁴ Hence, we can instead check the identifiability of $f_{\mathbf{s}}^i$ by invoking the ID-QFUNC subroutine (line 8). For example, suppose Figures 4.3(b) and 4.3(c) depict graphs T and H induced by contexts c_0e_0 and b_0c_0 . Both conditions hold for the c-component $\{E\}$ since E has the same parent C in the graphs and $(b_0c_0 \setminus c_0e_0)$ does not involve E or its parent. In contrast, condition (i) is violated for c-component $\{A, B, D\}$.

The ID-QFUNC checks the identifiability of c-factor $f_{\mathbf{s}}^i$ recursively. It initializes a variable set $\mathbf{D}$ containing all variables ($\mathbf{V} \cup \mathbf{U}$) and iteratively modifies it using the following two operations. Let $H_{\mathbf{D}}$ be the subgraph of H containing variables in $\mathbf{D}$ and all edge between them. At each step, either (1) remove a leaf node in $H_{\mathbf{D}}$ that is not in $\mathbf{S} \cup \mathbf{C}_i$ (lines 16-19); or (2) replace $\mathbf{D}$ by a maximal c-component $\mathbf{D}'$ in $H_{\mathbf{D}}$ such that $\mathbf{D} \supset \mathbf{D}' \supseteq \mathbf{C}_i$ (lines 20-22). If the recursion terminates with $\mathbf{D} = \mathbf{C}_i$, the c-factor is identifiable, and ID-QFUNC returns a formula for its estimation; otherwise, it returns FAIL. We formulate this c-component identification method as Lemma 6 in Appendix C.3.

We now illustrate how this identifiability method can be applied to treat the c-components in the CI-graph. Consider again the c-components $\{A, D, J\}$ and $\{S\}$ in the (mutilated) CI-graph in Figure 4.4(b) under the context $y_0j_0s_0$. To identify these c-components, let j_0 be the context $\mathbf{s}$ in line 6 and Figure 4.4(a) be its CI-graph H . Since $\mathbf{s}$ and H satisfy the conditions in line 7, the causal effect is identifiable if ID-QFUNC establishes identifiability for both c-components. This is indeed the case. The first c-component $\{A, D, J\}$ can be obtained by replacing $\mathbf{V}$ with the maximal c-component $\{A, D, J, Y\}$ and then pruning the leaf node Y . The second c-component $\{S\}$ is already a maximal c-component of T .

We now formally state the soundness of Algorithm 1.

Theorem 6. *Let $\mathcal{C}$ be a set of CSI constraints. Then $\text{Pr}_{\mathbf{x}}(\mathbf{y})$ is identifiable wrt $\langle G, \mathbf{V}, \mathcal{C} \rangle$ if*

⁴The two c-factors contain the same set of CPTs by condition (i). Moreover, if $\mathbf{s}$ contains the state of some variable in the CPTs, that state must appear in $\mathbf{xy}$ by condition (ii), which implies that the entries in $f_{\mathbf{s}}^i$ form a superset of those in $f_{\mathbf{xy}}^i$.

Algorithm 2 State-Based Identifiability with CSI and State Constraints

```
1: procedure ID-CSI-STATE( $G, \mathbf{V}, \mathbf{x}, \mathbf{y}, \mathcal{C}_1, \mathcal{C}_2$ )  
  // input: graph  $G$ , observed variables  $\mathbf{V}$ , treatment  $\mathbf{x}$ , outcome  $\mathbf{y}$ , CSI constraints  $\mathcal{C}_1$ , state constraints  $\mathcal{C}_2$   
  // output: identifying formula for  $\Pr_{\mathbf{x}}(\mathbf{y})$  or FAIL  
2:   for each possible context-partition  $\{\mathbf{s}_1, \dots, \mathbf{s}_k\}$  of  $\mathbf{V} \setminus (\mathbf{X} \cup \mathbf{Y})$  according to  $\mathcal{C}_2$  do  
3:     for  $i = 1$  to  $k$  do  
4:        $\Pr_{\mathbf{x}}(\mathbf{y}, \mathbf{s}_i) \leftarrow \text{ID-CSI}(G, \mathcal{C}_1, \mathbf{x}, \mathbf{y}, \mathbf{V})$   
5:       if  $\Pr_{\mathbf{x}}(\mathbf{y}, \mathbf{s}_i) \neq \text{FAIL}$  for  $i = 1, \dots, k$  then  
6:         return  $\sum_{i=1}^k \Pr_{\mathbf{x}}(\mathbf{y}, \mathbf{s}_i)$   
7:   return FAIL
```

Algorithm 1 returns a formula for estimating $\Pr_{\mathbf{x}}(\mathbf{y})$.

If the set of constraints $\mathcal{C}$ is empty, then Algorithm 1 is both sound and complete for testing identifiability. That is, when $\mathcal{C}$ is empty, Algorithm 1 can identify all causal effects that are identifiable by classical identification methods.

Proposition 5. $\Pr_{\mathbf{x}}(\mathbf{y})$ is identifiable wrt $\langle G, \mathbf{V} \rangle$ iff it is identifiable by Algorithm 1.

In general, for a causal graph with n variables, the time complexity of Algorithm 1 can be exponential in n , since the numbers of possible CI-graphs (line 2) and contexts (line 6) can be exponential in the worst case. If the total number of contexts appearing in CSIs is bounded by a constant, the worst-case complexity becomes polynomial.⁵

We have introduced a sound algorithm for testing state-based (variable-based) identifiability *when the only constraints are CSIs*. When the causal effect is identifiable, the algorithm also returns a formula for estimating the causal effect. We next discuss an extension of this algorithm that further enables leveraging state constraints in conjunction with CSIs.

4.4.2 Leveraging State Constraints

As demonstrated in Section 4.3, additional causal effects become identifiable when CSIs are coupled with state constraints. We next propose a systematic approach for testing state-based (and variable-based) identifiability under both CSI and state constraints by leveraging Algorithm 1. Note that we allow variables with arbitrary cardinalities, which is more general than [THK19] and [MJE22] which assume all variables are binary.

The detailed procedure is shown in Algorithm 2 which relies on an important notion called *context-partitions*. In particular, we say that a set $\{\mathbf{s}_1, \dots, \mathbf{s}_k\}$ forms a context-partition of variables $\mathbf{V}$ if (i) each $\mathbf{s}_i$, called a context, is an instantiation of some variables in $\mathbf{V}$, (ii) $\mathbf{s}_i$ and $\mathbf{s}_j$ are not compatible for $i \neq j$, and (iii) every instantiation $\mathbf{v}$ of $\mathbf{V}$ is compatible with some context $\mathbf{s}_i$. That is, contexts do not overlap and cover all instantiations of $\mathbf{V}$. To illustrate, suppose variables A, B, C have states $\{a_0, a_1, a_2\}$, $\{b_0, b_1\}$ and $\{c_0, c_1\}$, respectively. Then $\{a_0, a_1b_0, a_1b_1, a_2c_1, a_2c_2\}$ is a context-partition of these variables and so is $\{a_0b_0, a_0b_1, a_1, a_2b_0c_0, a_2b_0c_1, a_2b_1\}$.

Algorithm 2 enumerates all possible context-partitions $\{\mathbf{s}_1, \dots, \mathbf{s}_k\}$ over the variable states specified by state constraints, and reduces the problem of identifying $\Pr_{\mathbf{x}}(\mathbf{y})$ to that of identifying sub-queries $\Pr_{\mathbf{x}}(\mathbf{y}, \mathbf{s}_i)$ for each context $\mathbf{s}_i$ (line 2). It then invokes Algorithm 1 as a subroutine to check the identifiability of these sub-queries (lines 3-6). If all sub-queries are deemed identifiable, $\Pr_{\mathbf{x}}(\mathbf{y})$ is also identifiable. The intuition is that, by augmenting the treatment and outcome states $\mathbf{xy}$ with additional contexts $\mathbf{s}_i$, we can exploit additional CSIs whose contexts extend beyond $\mathbf{xy}$ and hence identify more causal effects. When the causal effect is identifiable, the algorithm also returns a formula for estimating it using the equality $\Pr_{\mathbf{x}}(\mathbf{y}) = \sum_i \Pr(\mathbf{y}, \mathbf{s}_i)$.

We state the soundness of Algorithm 2 for identifiability under CSI and state constraints.

⁵It suffices to enumerate $\mathbf{s}$ for which all states appear in the contexts of CSIs in line 6 without compromising the completeness. The complexity is dominated by ID-QFUNC in this case.

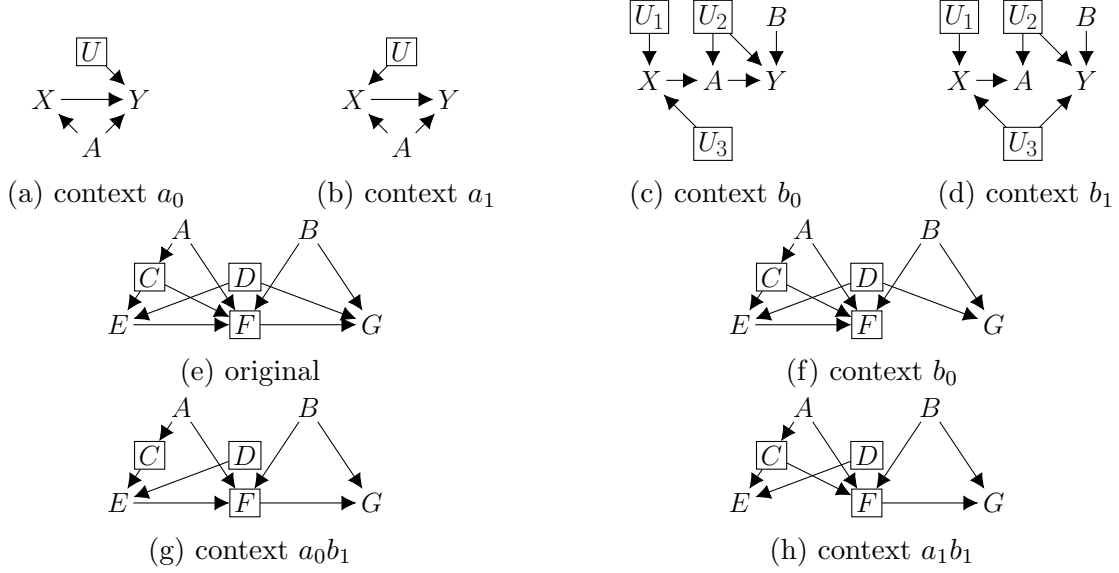


Figure 4.5: CI-graphs under different contexts in the context-partition for various examples.

Theorem 7. *Let $\mathcal{C}_1$ be a set of CSIs and $\mathcal{C}_2$ a set of state constraints. Then $\Pr_{\mathbf{x}}(\mathbf{y})$ is identifiable wrt $\langle G, \mathbf{V}, \mathcal{C}_1 \cup \mathcal{C}_2 \rangle$ if Algorithm 2 returns a formula for estimating $\Pr_{\mathbf{x}}(\mathbf{y})$.*

We next demonstrate how Algorithm 2 can be applied to identify causal effects when state constraints are available. First consider the causal graph in Figure 4.2(a) with the CSI constraints $(X \perp\!\!\!\perp U \mid a_0)$, $(Y \perp\!\!\!\perp U \mid X, a_1)$. Suppose variable A only has two states $\{a_0, a_1\}$. Instead of directly checking the identifiability of the original $\Pr_{x_0}(y_0)$, we split the causal effect into sub-queries $\Pr_{x_0}(y_0, a_0)$ and $\Pr_{x_0}(y_0, a_1)$ according to the context-partition $\{a_0, a_1\}$. Figures 4.5(a) and 4.5(b) depict the CI-graphs for the sub-queries. Since both sub-queries are identifiable by Algorithm 1, the original causal effect is also identifiable; see the proof of Proposition 19 in Appendix C.3. Similarly, we can apply the theorem to identify $\Pr_{x_0}(y_0)$ in Figure 4.2(b) by constructing two sub-queries whose CI-graphs are shown in Figures 4.5(c) and 4.5(d); see the proof of Proposition 20 in Appendix C.3 for more details.

We now show another example with the causal graph in Figure 4.5(e) that requires splitting more than one variable in the context-partition. Suppose the states for variables A, B are $\{a_0, a_1\}$, $\{b_0, b_1\}$ and assume the following CSI constraints: $(G \perp\!\!\!\perp F \mid b_0, D)$, $(G \perp\!\!\!\perp$

$\perp D \mid b_1, F), (F \perp\!\!\!\perp C \mid a_0, B, E), (F \perp\!\!\!\perp E \mid a_1, B, C)$. The state-based causal effect $\Pr_{e_0}(g_0)$ (and variable-based $\Pr_E(G)$) is identifiable since each sub-query induced by the context-partition $\{b_0, a_0b_1, a_1b_1\}$ can be identified using Algorithm 1; see Figures 4.5(f) to 4.5(h) for the CI-graphs under each context in the partition.

We next discuss a method for testing *variable-based* identifiability under both CSI and state constraints. Similar to the case with CSI constraints only (Proposition 4 in Section 4.2), the method is based on reducing variable-based identifiability to state-based identifiability. Specifically, to check the identifiability of $\Pr_{\mathbf{X}}(\mathbf{Y})$, we consider instantiations $\mathbf{x}, \mathbf{y}$ in which each state is either included in the state constraints or outside the CSI contexts. The number of such instantiations is finite if every variable outside the state constraints is assigned the same state (outside the CSI contexts) across all instantiations. Then $\Pr_{\mathbf{X}}(\mathbf{Y})$ is identifiable iff $\Pr_{\mathbf{x}}(\mathbf{y})$ is identifiable for all these instantiations. We formalize this result as Proposition 21 in Appendix C.3.

4.4.3 Heuristics

Enumerating CI-graphs (Algorithm 1, line 2), contexts and their corresponding CI-graphs (Algorithm 1, lines 6-7), and context-partitions (Algorithm 2 line 2) incur exponential time complexity and are therefore impractical. To address this issue, we propose in Appendix C.2 heuristics to select a single choice for each step, avoiding the need to enumerate all possible choices. Algorithms 1 and 2 combined with these heuristics will be denoted by ID-CSI-HEU and ID-CSI-STATE-HEU. We will use these versions of the algorithms in our upcoming experiments which demonstrate that the heuristics substantially improve the computational efficiency of our proposed identification algorithms.

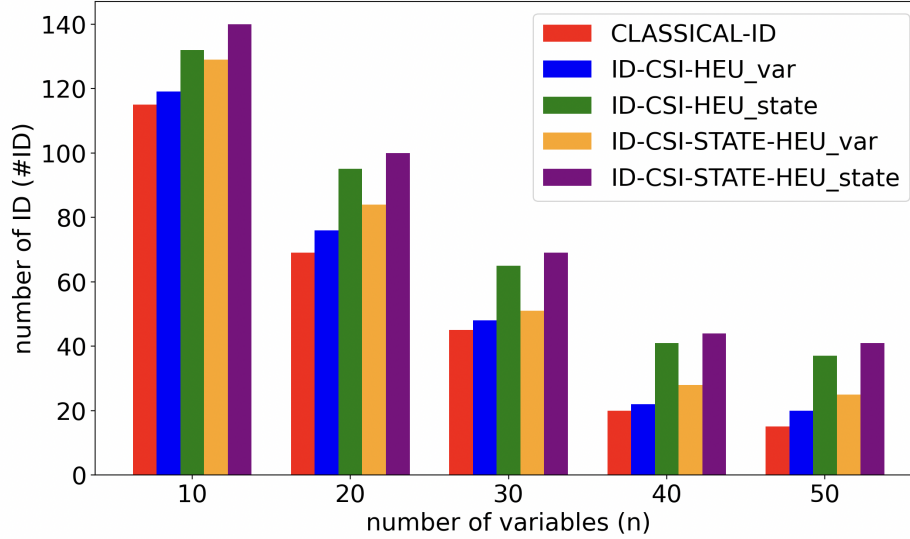


Figure 4.6: Number of identifiable causal effects (out of 200 runs) as the number of variables increases with $r = 0.8$. Methods are labeled with the suffix “_var” for variable-based identifiability and “_state” for state-based identifiability. The two levels of identifiability are equivalent for CLASSICAL-ID.

4.5 Experiments

We will now present empirical results illustrating the improvement in both variable-based and state-based identifiability due to CSI and state constraints, and further highlighting the separations between the two levels of identifiability.

We consider the following setups of identifiability in our experiments: (1) without constraints (CLASSICAL-ID), (2) with CSI constraints only (ID-CSI-HEU), and (3) with both CSIs and state constraints, assuming all observed variables are binary (ID-CSI-STATE-HEU). For all setups, we evaluate identifiability at both the variable-based and state-based levels. Note that the two levels are equivalent in setup (1) based on Proposition 3. For setup (3), we also compare with the DOSEARCH package [THK19], which targets variable-based identifiability for binary variables only.⁶

For all sets of experiments, we randomly generate 200 causal graphs that contain $n \in$

⁶The DOSEARCH package is available at <https://github.com/santikka/dosearch>.

$\{6, 8, 10, 12, 15, 20, 30, 40, 50\}$ variables. Each variable has at most 5 parents. Among the variables, 70% are designated to be observed, 20% as treatment, and 20% as outcome. The CSI constraints are randomly generated as follows. For each variable C with parent P_1 , we assign a CSI $(C \perp\!\!\!\perp P_1 \mid \mathbf{p}_2, \mathbf{P}_3)$ with probability $r \in \{0.2, 0.5, 0.8\}$. We then assign a random state of other parents (excluding P_1) to $\mathbf{p}_2$ with probability 0.5. Note that larger values of r correspond to more imposed CSIs.

Table C.1 in Appendix C.4 reports the average number of identifiable causal effects and runtime for each method, with a 5-minute timeout for each run. Figure 4.6 plots the number of identifiable causal effects with an increasing number of variables n and a fixed $r = 0.8$. The following patterns can be observed. First, for both variable-based and state-based identifiability, more causal effects become identifiable when additional CSI constraints are imposed (i.e., larger r) and when state constraints are present, which matches our intuition that constraints can facilitate identifiability. Second, the gap between state-based and variable-based identifiability widens as more CSI constraints are imposed. For example, when $n=50$ and state constraints are present, the gap in the number of identifiable causal effects increases from 4 to 16 when r increases from 0.2 to 0.8. A similar trend holds in the absence of state constraints. Finally, our algorithm ID-CSI-STATE-HEU identifies a comparable – sometimes larger – number of variable-based causal effects compared to DOSEARCH, while achieving significantly higher computational efficiency (DOSEARCH times out on the majority of runs when $n \geq 15$). These results demonstrate the effectiveness of our proposed algorithms when applied to both levels of identifiability.

It is worth emphasizing that our algorithms adopt a graph decomposition approach using c-components, in contrast to the exhaustive search method based on CSI-calculus rules introduced in [THK19] and used in DOSEARCH. In fact, one may draw an interesting analogy here: our algorithms relate to CSI-calculus in a similar way to how IDENTIFY [TP03] relate to the do-calculus [pea09] as noted in [THK19]. A key advantage of our procedural approach is that it makes the identification process more explicit than exhaustive search, enabling

computational optimizations (e.g., the heuristics developed in Section 4.4.3) and facilitating future analysis of completeness.

CHAPTER 5

Testing Constrained Identifiability Using Tractable Arithmetic Circuits

We have so far presented methods for identifying causal effects under the knowledge (constraints) of functional dependencies and context-specific independencies. In this chapter, we introduce a different approach for testing constrained identifiability based on tractable Arithmetic Circuits (ACs), a data structure representing a particular computation procedure and has played an influential role in probabilistic inference [Dar01]. We start with a background on ACs in Section 5.1 and describe the procedure for identifying causal effects in Sections 5.2 and 5.3. The use of ACs allows us to systematically incorporate constraints on model parameters, such as parameter equalities and known values. Another interesting setup is when the observational distribution $\Pr(\mathbf{V})$ is fully known, in which case all parameters involving observed variables become known and can therefore be fixed. The improvement enabled by leveraging the constraints via ACs is further demonstrated through examples in Section 5.4. The results in this chapter appeared in [CD25a].¹

5.1 Background on Tractable Arithmetic Circuits (ACs)

We first review the notion of tractable arithmetic circuits (ACs) [Dar01, Dar03, Dar09, Dar21b]. In essence, an AC is a data structure representing a particular computation pro-

¹To maintain consistency with other chapters, some definitions in this chapter differ slightly from those in [CD25a]. For example, we define constrained identifiability in terms of the constraints $\mathcal{C}$ rather than the set of models satisfying $\mathcal{C}$, as done in [CD25a]. These are minor changes that do not affect the results.

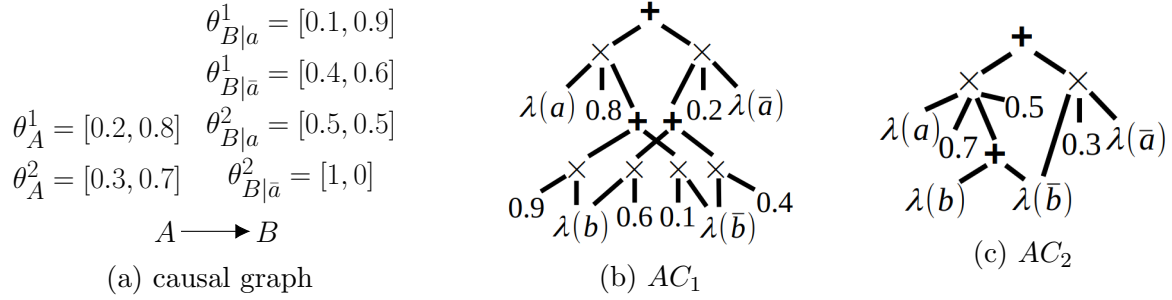


Figure 5.1: Arithmetic circuits for two different models.

cedure. By definition, an AC is a rooted DAG containing four types of nodes: parameters (θ 's), indicators (λ 's), $+$ -nodes, and $\times$ -nodes, where parameters and indicators serve as leaf nodes, and the arithmetic gates serve as internal nodes. For example, Figures 5.1(b) and (c) depict the ACs for the graph in Figure 5.1(a) under different parameter sets Θ^1 and Θ^2 .

When an AC is compiled from a probabilistic model (e.g., a causal Bayesian network), the parameters (θ 's) in the AC are typically derived from the model's parameters so that they can be used for probabilistic inference tasks, such as computing marginals; see [Dar09] for details on AC properties and the types of inferences they support. Knowledge compilation also allows us to exploit parameter-specific constraints (local structure), such as equal parameters (e.g., context-specific independences) and 0/1 parameters (e.g., logical constraints), which leads to simpler ACs; see, e.g., [Dar02, CD07, HCA23, HD24]. To illustrate, the AC for (G, Θ^2) in Figure 5.1(c) contains fewer nodes than the AC for (G, Θ^1) in Figure 5.1(b) since Θ^2 exhibits more constraints that get exploited during AC construction. As we will see later, this can lead to more complete causal effect identifications using the method we propose.

5.2 Procedure for Testing Identifiability Using ACs

Our method for testing identifiability using ACs involves two steps (ingredients): *AC Construction* and *Invariance Testing*. The first step constructs an AC whose output computes the causal effect $\Pr_{\mathbf{x}}(\mathbf{y})$ and which incorporates the constraints. The second step tests identi-

fiability by checking whether the output of the AC is invariant under all models that satisfy the constraints ($\mathcal{C}$ in Definition 3). Hence, the main objective of this and future work is to improve the quality of these two steps so that we obtain a more complete method for identifying causal effects under constraints. Throughout this chapter, we will assume that the states of all variables (including hidden ones) are known, and we will point out in case this assumption can be omitted.²

5.2.1 AC Construction

We start by constructing an AC for the causal effect $\Pr_{\mathbf{x}}(\mathbf{y})$ based on the approach in [Dar21a]. First, compile the original BN into an AC in a standard way, with indicators λ_v for each variable state v and parameters $\theta_{v|\mathbf{p}}$ for each instantiation over V and its parents $\mathbf{P}$.³ Second, replace every parameter $\theta_{x|\mathbf{p}}$ with 1 for each treatment $X \in \mathbf{X}$. Finally, for each $V \in \mathbf{X} \cup \mathbf{Y}$, assign $\lambda_v = 1$ if v is compatible with $\mathbf{x}, \mathbf{y}$ and $\lambda_v = 0$ otherwise; assign 1 to all other indicators (this is how we estimate an AC under evidence $\mathbf{x}, \mathbf{y}$).

To evaluate an AC under a model M , we simply plugin each (non-constant) parameter $\theta_{v|\mathbf{p}}$ with the value from M and compute the output of the AC. A key observation is that the ACs constructed from the above procedure are guaranteed to compute $\Pr_{\mathbf{x}}(\mathbf{y})$ correctly under *all* parameterizations of G . As we will see later, when additional constraints are available, the ACs may be further simplified, leading to more complete identifiability results in practice. In general, the leaves of an AC are either constants (e.g., 0.8) or parameters ($\theta_{v|\mathbf{p}}$).

Consider the causal graph G in Figure 5.2(a) with observed variables $\mathbf{V} = \{X, A, B, Y\}$. Assuming all variables are binary, Figure 5.2(b) depicts an AC for $\Pr_x(y)$. Due to the space limit, we only plotted one branch for most $+$ -nodes in the figure. In general, we prefer simpler

²[ZTB22] showed that all counterfactual distributions (which subsumes observational and interventional distributions) can be captured by causal models whose hidden variables have a *bounded* number of states.

³Note that each $\theta_{v|\mathbf{p}}$ is equivalent to the conditional probability $\Pr(v|\mathbf{p})$ when $\Pr(\mathbf{p}) > 0$. Hence, we will use $\theta_{v|\mathbf{p}}$ and $\Pr(v|\mathbf{p})$ interchangeably when the positivity condition holds.

(smaller) ACs since they can sometimes lead to more complete causal effect identifiability. While most AC simplifications can be handled automatically by the **ACE** package,⁴ we propose additional rules that can be used to further simplify ACs in certain scenarios:

1. (Sum-out) For a $+$ -node in the AC, if it computes $\sum_v \theta_{v|\mathbf{p}}$, replace the $+$ -node with constant 1.
2. (Merge) If two nodes in the AC are of the same type and have the same set of children, merge them into a single node.
3. (Distributivity) For a $+$ -node i whose children $j_1, \dots, j_n$ are $\times$ -nodes sharing a common child k : remove k from the children of $j_1, \dots, j_n$, and replace i with a $\times$ -node whose children are k and a new $+$ -node with children $j_1, \dots, j_n$.

These rules can be applied recursively to simplify the AC to the maximum extent. Note that we can always omit AC nodes that evaluate to zero and nodes that evaluate to one if their parent is a $\times$ -node; we illustrate the application of these rules with examples later.

5.2.2 Invariance Testing

The AC constructed from the previous step is guaranteed to compute the causal effect under all models for the causal graph. We next develop a method for testing identifiability based on the following observation: a causal effect $\Pr_{\mathbf{x}}(\mathbf{y})$ is (constrained) identifiable wrt $\langle G, \mathbf{V}, \mathcal{C} \rangle$ if the output of the ACs for $\Pr_{\mathbf{x}}(\mathbf{y})$ is *invariant* under all models that satisfy $\mathcal{C}$ and induce a same $\Pr(\mathbf{V})$.

In this work, an *expression* is defined as a composition of constants (e.g., 0.8), parameters ($\theta_{v|\mathbf{p}}$), and conditional probabilities (e.g., $\Pr(a|b)$). Note that each parameter $\theta_{v|\mathbf{p}}$ is equal to the conditional probability $\Pr(v|\mathbf{p})$ when $\Pr(\mathbf{p}) > 0$. For example, $\Pr(y|x)$, 0.65,

⁴<http://reasoning.cs.ucla.edu/ace>.

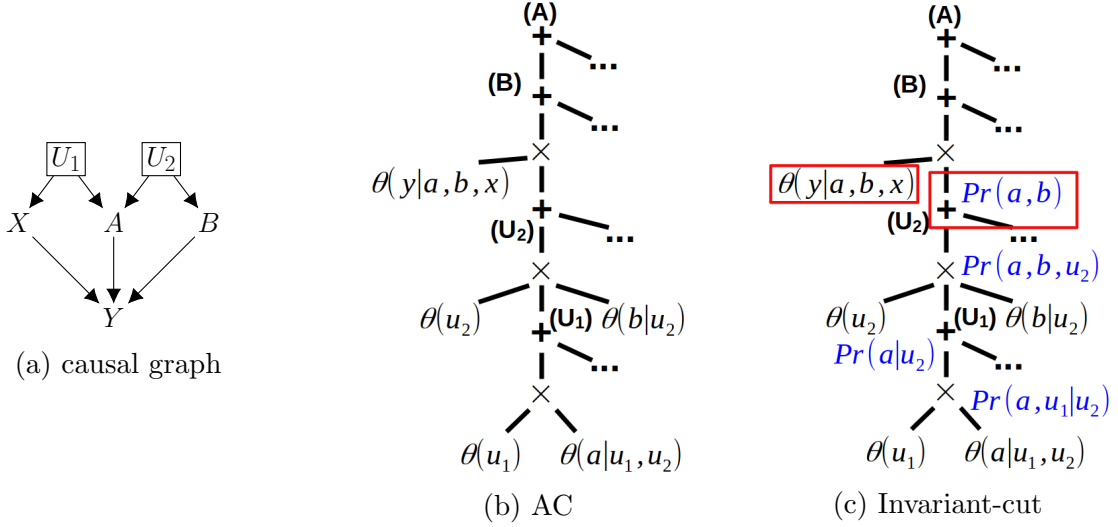


Figure 5.2: AC constructed for $\Pr_x(y)$ for a causal graph; only one branch for each $+$ -node is plotted; equivalent expressions are shown in blue and nodes in the invariant-cut are marked with red boxes.

$0.2 \Pr(a|x) + 0.6 \Pr(x)$, $\sum_u \Pr(a|x) \Pr(u)$, $\theta(a|x) \Pr(x) + 0.3 \Pr(\bar{a})$ are all valid expressions. An AC specifies an expression consisting of $\times$'s and $+$'s, which can be obtained by evaluating the output of the AC in a standard (bottom-up) way. For example, the AC in Figure 5.2(b) specifies the expression $\sum_a \sum_b \theta_{y|a,b,x} \sum_{u_2} \theta_{u_2} \theta_{b|u_2} \sum_{u_1} \theta_{u_1} \theta_{a|u_1,u_2}$. Moreover, every node in an AC corresponds to an expression so we will use “AC node” and “expression” interchangeably.

Our next goal is to propose a method for testing identifiability that operates on ACs. We start by defining some key notions that are required to achieve this goal.

Definition 10. *An expression is called $\langle G, \mathbf{V}, \mathcal{C} \rangle$ -invariant if it evaluates to the same value for all models that satisfy $\mathcal{C}$ and induce a same $\Pr(\mathbf{V})$.*

The following result shows that we can test identifiability by checking whether the output of an AC is invariant.

Proposition 6. *A causal effect $\Pr_{\mathbf{x}}(\mathbf{y})$ is identifiable wrt $\langle G, \mathbf{V}, \mathcal{C} \rangle$ iff there is an AC for $\Pr_{\mathbf{x}}(\mathbf{y})$ whose output expression is $\langle G, \mathbf{V}, \mathcal{C} \rangle$ -invariant.*

In general, it is difficult to tell whether the output of an AC is invariant since the

evaluation of the AC often contains hidden variables $\mathbf{U}$. However, if we are able to replace the expressions for some AC nodes with *equivalent expressions* that do not depend on $\mathbf{U}$, we may obtain output expressions that are obviously invariant, e.g., expressions that only involve observed variables $\mathbf{V}$.

Definition 11. *Two expressions $\mathcal{E}_1$ and $\mathcal{E}_2$ are called $\langle G, \mathbf{V}, \mathcal{C} \rangle$ -equivalent if they evaluate to the same value under all models satisfying $\mathcal{C}$. If in addition $\mathcal{E}_1$ does not contain any parameters (θ 's) and does not mention any hidden variables,⁵ it is called an $\langle G, \mathbf{V}, \mathcal{C} \rangle$ -projection of $\mathcal{E}_2$.*

The following theorem presents a method for testing identifiability.

Proposition 7. *The causal effect $\Pr_{\mathbf{x}}(\mathbf{y})$ is identifiable wrt $\langle G, \mathbf{V}, \mathcal{C} \rangle$ if there is an AC for $\Pr_{\mathbf{x}}(\mathbf{y})$ that satisfies the following condition: there is a cut between the root and leaf nodes where every node on the cut has a $\langle G, \mathbf{V}, \mathcal{C} \rangle$ -projection.*

We call the cut in Proposition 7 an *invariant-cut*. Figure 5.2(c) depicts the projections (in blue) and an invariant-cut (in red boxes) for the AC in Figure 5.2(b). Hence, we conclude that the causal effect $\Pr_x(y)$ is identifiable. Once we attain an invariant-cut we immediately obtain an identifying formula by plugging in the projections for the nodes on the invariant-cut and evaluating the AC root. In this example, the identifying formula for the causal effect is $\Pr_x(y) = \Pr(y|x, a, b)\Pr(a, b) + \Pr(y|x, a, \bar{b})\Pr(a, \bar{b}) + \Pr(y|x, \bar{a}, b)\Pr(\bar{a}, b) + \Pr(y|x, \bar{a}, \bar{b})\Pr(\bar{a}, \bar{b})$, which resembles the backdoor adjustment formula. Invariant-cuts may not be unique as the choice of projections is not unique. We may also have multiple ACs with invariant-cuts, which further contributes to the multiplicity of identifying formulas.

Another interesting observation is that the root of the AC always constitutes a valid invariant-cut when one exists. However, since lower cuts involve smaller expressions, finding efficient projections for nodes on such cuts may be easier, as we show next.

⁵The observational distribution $\Pr(\mathbf{V})$ does not depend on the value of parameter $\theta_{x|\mathbf{p}}$ when $\Pr(\mathbf{p}) = 0$. However, the interventional distribution $\Pr_{\mathbf{x}}(\cdot)$ and, hence, causal effects may depend on the value of $\theta_{x|\mathbf{p}}$. This is why we require projections to be free of parameters as they may not be computable from $\Pr(\mathbf{V})$. However, since $\theta_{x|\mathbf{p}} = \Pr(x|\mathbf{p})$ when $\Pr(\mathbf{p}) > 0$ we can always replace $\theta_{x|\mathbf{p}}$ with $\Pr(x|\mathbf{p})$ when $\Pr(\mathbf{p}) > 0$.

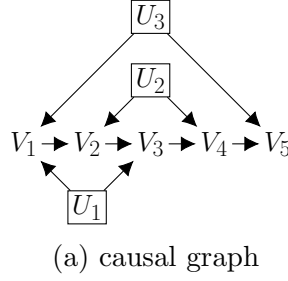


Figure 5.3: A causal graph from [TP02b]. Hidden variables are indicated with boxes.

5.3 Finding Invariant-Cuts For ACs

Our method for finding projections for ACs is based on the notion of c-components, which has been studied extensively in [TP02b, TP03, SP06, HV06] and reviewed in Appendix A.2. Intuitively, a (maximal) c-component is a (maximal) subset of variables in the causal graph in which all observed variables are connected through one or more shared hidden confounders; see Definition 19 in Appendix A.2. Moreover, each c-component $\mathbf{S}$ induces a *c-factor* defined as $Q[\mathbf{S}] = \sum_{\mathbf{H}} \prod_{S \in \mathbf{S}} \theta_{S|\mathbf{P}_S}$, where $\mathbf{H}$ denotes the hidden variables in $\mathbf{S}$ and $\mathbf{P}_S$ denotes the parents of S in the causal graph G . As noted earlier, our definition of c-component also includes hidden variables, which is more explicit than the one in [TP02b].

Existing methods in [TP03] identify causal effects by decomposing the mutilated graph G' under $do(\mathbf{x})$ into c-components and then checking whether all corresponding c-factors have a valid projection. We next present the notion of **V-computable** variable sets for checking the existence of projections, which is equivalent to the key results in [TP03, HV06, TP02b].

Definition 12. *Let G be a causal graph with observed variables $\mathbf{V}$. A variable set is called V-computable if it can be obtained from $\mathbf{Z} = (\mathbf{V} \cup \mathbf{U})$ by repeatedly applying the following operations: (1) Replace $\mathbf{Z}$ with a maximal c-component of $G_{\mathbf{Z}}$; or (2) Remove a leaf node in $G_{\mathbf{Z}}$, where $G_{\mathbf{Z}}$ denotes the subgraph of G formed by $\mathbf{Z}$.⁶*

⁶In this dissertation, the property of **V-computability** is defined for variable sets, which differs from the original definition for subgraphs in [CD25a]. We adopt this modified definition here since it is more consistent with the discussion in the previous chapters. This does not affect the validity of the results.

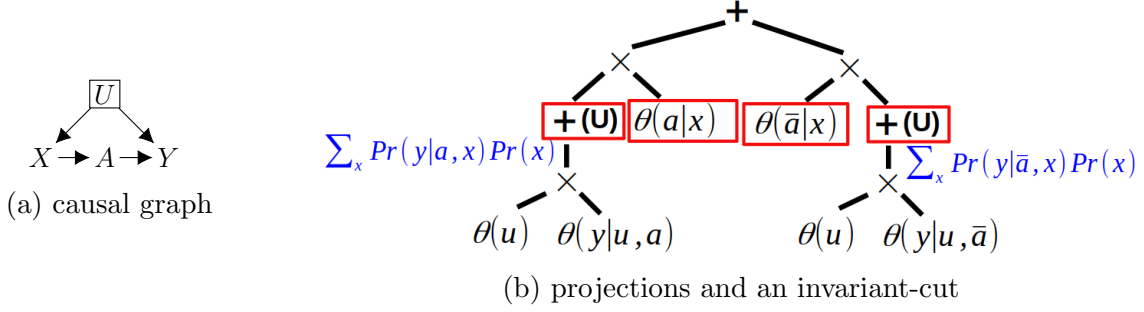


Figure 5.4: An AC constructed for $\Pr_x(y)$; an invariant-cut (in red boxes) is found by the c-component method.

Corollary 5. *Let G be a causal graph with observed variables $\mathbf{V}$. If a c-component $\mathbf{S}$ is $\mathbf{V}$ -computable, then $Q[\mathbf{S}]$ has a $\langle G, \mathbf{V}, \Pr(\mathbf{V}) > 0 \rangle$ -projection.*

The IDENTIFY method in [TP03] further derives a concrete expression for the projection whenever it exists. To illustrate, the c-component $\mathbf{S} = \{U_1, U_3, V_1, V_5\}$ in Figure 5.3(a) is $\mathbf{V}$ -computable, as it can be obtained by taking the c-component $\{U_1, U_3, V_1, V_3, V_5\}$ and pruning the leaf V_3 . Moreover, it induces the following c-factor: $Q[\mathbf{S}] = \sum_{U_1, U_3} \theta_{U_1} \theta_{U_3} \theta_{V_1|U_1, U_3} \theta_{V_5|U_3, V_4}$, which has a valid projection under positivity $\Pr(\mathbf{V}) > 0$ by Corollary 5. Now suppose the expression at some AC node i matches the c-factor, we can replace it with the corresponding projection which only involves $\mathbf{V}$.

Consider the causal graph in Figure 5.4(a) with binary observed variables $\mathbf{V} = \{X, Y, A\}$ and assume $\Pr(\mathbf{V}) > 0$. Figure 5.4(b) depicts an AC constructed for the causal effect $\Pr_x(y)$. Since the set $\{U, Y\}$ is $\mathbf{V}$ -computable, we can obtain the following c-factor $Q[UY] = \sum_U \theta_U \theta_{Y|U, A}$, which matches exactly the expression at the $+$ -node on the left branch of the AC. Hence, we can assign the corresponding projection for the $+$ -node as shown in Figure 5.4(b). It turns out that the causal effect is identifiable since we can find an invariant-cut (in red boxed) for the AC.

The following theorem shows that the *c-component method* in Corollary 5, when coupled with carefully designed ACs, is sound and complete for testing identifiability under $\Pr(\mathbf{V}) > 0$, matching the classical identification methods such as IDENTIFY [TP03] and ID [SP06].

Theorem 8. *A causal effect $\Pr_{\mathbf{x}}(\mathbf{y})$ is identifiable wrt $\langle G, \mathbf{V}, \Pr(\mathbf{V}) > 0 \rangle$ iff there is an AC for $\Pr_{\mathbf{x}}(\mathbf{y})$ on which the c-component method in Corollary 5 finds an invariant-cut.⁷*

When additional constraints such as context-specific independencies (CSIs) are available, we may improve the discovery of projections using the ID-QFUNC function in Algorithm 1, allowing more causal effects to be identified, which we will illustrate with examples later.

5.4 Exploiting Constraints in Identifiability: Examples

We next use examples to demonstrate how our AC-based method can be applied to improve causal effect identifiability under different types of constraints. In particular, we consider the constraints of context-specific independencies (CSIs) and a fully-known observational distribution $\Pr(\mathbf{V})$, and show that we can exploit these constraints to simplify the constructed ACs in order to facilitate the discovery of invariant-cuts. While the use of CSIs has been explored in Chapter 4, the AC-based method presented here offers a more systematic treatment and enables integration with other types of knowledge, such as known parameters. It is also worth noting that the method for incorporating functional dependencies in Chapter 3 can be used as a *preprocessing mechanism* that reduces identifiability problems with functional variables to ones without them; the method remains sound when variable states are known.

All the ACs considered in this section are compiled using the variable elimination method discussed in [CD07], with constrained variable elimination orders: first eliminate all hidden variables $\mathbf{U}$, then observed variables $\mathbf{V}$ in a bottom-up order. These ACs may be further simplified using the simplification rules in Section 5.2.1.

⁷We assume all variables are binary when constructing the AC.

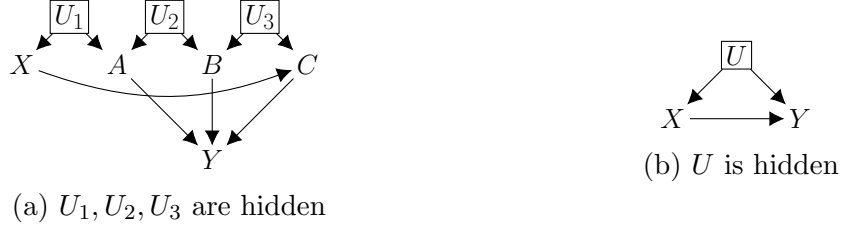


Figure 5.5: Causal graphs that exhibit different behaviors of identifiability for $\Pr_x(y)$.

5.4.1 Context-Specific Independencies

CSI constraints can be incorporated into the construction of ACs as follows. For each CSI relation $(X \perp\!\!\!\perp P_1 \mid \mathbf{p}_2, \mathbf{p}_3)$, we merge all the parameters $\theta_{x|p_1, \mathbf{p}_2, \mathbf{p}_3}$ (which are leaves) in the AC into a single node $\theta_{x|\mathbf{p}_2, \mathbf{p}_3}$. This is valid since $\theta_{x|p_1, \mathbf{p}_2, \mathbf{p}_3} = \theta_{x|p'_1, \mathbf{p}_2, \mathbf{p}_3}$ for all states p_1 and p'_1 [Dar02]. We can then apply simplification rules, such as those provided in Section 5.2.1, to facilitate the discovery of invariant-cuts.

Consider the causal graph in Figure 5.5(a) where the causal effect $\Pr_x(y)$ is not identifiable under the positivity constraint $\Pr(\mathbf{V}) > 0$. Assume now the following CSI constraints $\mathcal{C}_{csi}$: $(Y \perp\!\!\!\perp C \mid a, B)$ and $(Y \perp\!\!\!\perp B \mid \bar{a}, C)$. Figure 5.6(a) depicts the original AC for $\Pr_x(y)$ under these CSI constraints after merging $\theta_{y|a,b,c}$ and $\theta_{y|a,b,\bar{c}}$ into $\theta_{y|a,b}$, and merging $\theta_{y|\bar{a},b,c}$ and $\theta_{y|\bar{a},\bar{b},c}$ into $\theta_{y|\bar{a},c}$ (resulting nodes are colored in blue). We can simplify the AC by applying the simplification rules as follows. First, we push the $+$ -nodes downward (distributivity rule) and sum out the leaf nodes (sum-out rule); the removed $+$ -nodes and leaf nodes are highlighted in red in Figure 5.6(a). We then apply the distributivity rule on $+(U_3)$ -node on the right branch, yielding the simplified AC in Figure 5.6(b).

Given the simplified AC, we can then find an invariant-cut using the c-component method in Corollary 5. The projections for parameters such as $\theta_{y|a,b}$ and $\theta_{y|\bar{a},c}$ are directly $\Pr(y|a, b)$ and $\Pr(y|\bar{a}, c)$, since they involve only observed variables. The projections $\Pr(a, b)$, $\Pr(\bar{a})$, and $\Pr(c|x)$ for the remaining $+$ -nodes can be identified from the c-factors $Q[U_1 U_2 U_3 A B]$, $Q[U_1 U_2 A]$, and $Q[U_3 C]$, respectively, as their corresponding c-components are $\mathbf{V}$ -computable by Definition 12. Hence, we conclude that the causal effect is identifiable

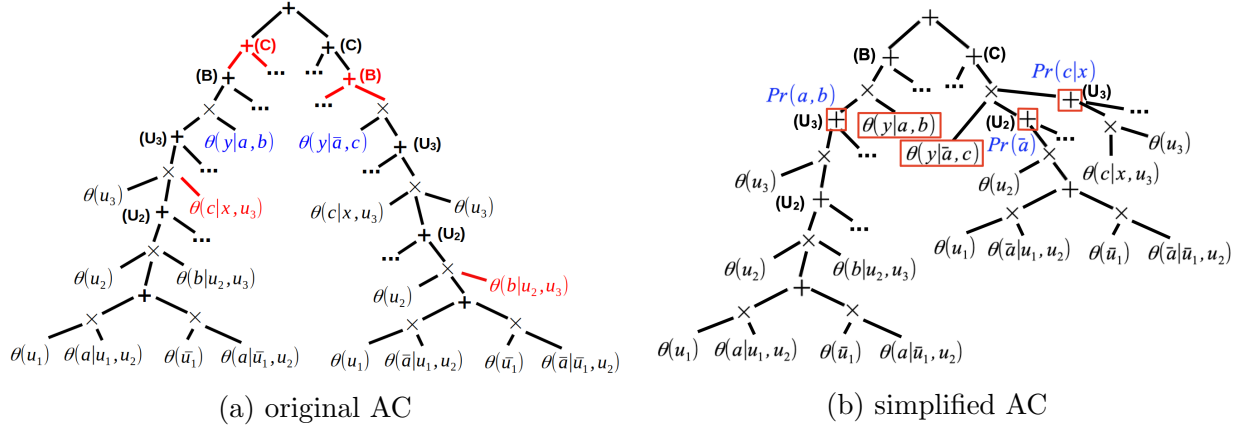


Figure 5.6: AC for Figure 5.5(a) with projections derived with the c-component method; projections are marked near the nodes in blue; the invariant-cut is marked with red boxes.

wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{csi}, \Pr(\mathbf{V}) > 0\} \rangle$. Plugging in these projections yields the following identifying formula: $\Pr_x(y) = \Pr(a, y) + \sum_c \Pr(y|\bar{a}, c) \Pr(c|x) \Pr(\bar{a})$.

We now revisit the example shown in Chapter 4 Figure 4.2(a). Assume all variables are binary and that the CSI constraints $(X \perp\!\!\!\perp U \mid \bar{a})$ and $(Y \perp\!\!\!\perp U \mid a, X)$ hold. Figure 5.7(c) depicts a simplified AC for the causal effect $\Pr_x(y)$. In this case, the method in Corollary 5 can no longer find an invariant-cut since it fails to derive a projection for the $+$ -node (in the red box), even though such a cut exists. To address this, we consider the refined c-component identification method ID-QFUNC from Algorithm 1 (Chapter 4), which further accounts for the CSI constraints. Specifically, using the context $\mathbf{s} = \bar{a}$ and the corresponding context-induced (CI-) graph H in Figure 5.7(b), we can identify the c-factor $Q[UY]$ and obtain the projection $\Pr(y|x, \bar{a})$ for the $+$ -node. Plugging in the projection for all nodes on the invariant-cut yields the following identifying formula: $\Pr_x(y) = \Pr(y|x, a) \Pr(a) + \Pr(\bar{a}) \Pr(y|x, \bar{a})$.

5.4.2 Fully-Known Observational Distributions

Another type of constrained identifiability arises when we restrict the observational distribution to a fixed $\Pr^*(\mathbf{V})$. That is, we only consider models that induce the given $\Pr^*(\mathbf{V})$. This is quite common in practice when data over observed variables $\mathbf{V}$ allows us to estimate

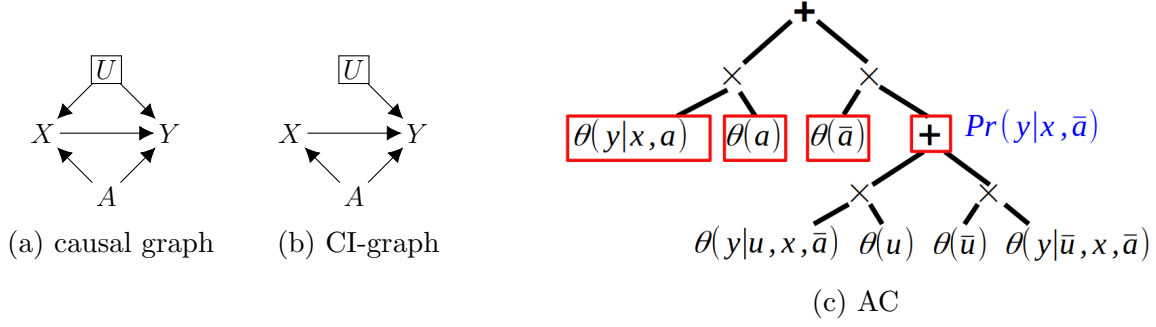


Figure 5.7: CSI example from Chapter 4 and an AC with an invariant-cut for $\Pr_x(y)$.

$\Pr^*(\mathbf{V})$ accurately. To illustrate, consider the diagram in Figure 5.8, which contains the set of all models induced by a causal graph. We can partition the set of models into *equivalence classes*, where all models in a particular class induce the same distribution $\Pr(\mathbf{V})$ — hence, an equivalence class is defined by its distribution $\Pr(\mathbf{V})$.⁸ In classical identifiability, to show that a causal effect is identifiable, we need to show that all models in the same equivalence class produce the same causal effect $\Pr_x(y)$. When $\Pr^*(\mathbf{V})$ is fixed, however, we only consider a single equivalence class $\Pr^*(\mathbf{V})$ and check whether all models in this particular class produce the same value for the causal effect $\Pr_x(y)$. We denote the identifiability tuple under a fixed $\Pr^*(\mathbf{V})$ as $\langle G, \mathbf{V}, \Pr(\mathbf{V}) = \Pr^*(\mathbf{V}) \rangle$.

A causal effect that is not identifiable by Definition 1 under $\Pr(\mathbf{V}) > 0$ may still be identifiable under *some* particular $\Pr^*(\mathbf{V}) > 0$. This happens when we have two distinct equivalence classes C_1 and C_2 where the models in C_1 agree on the causal effect but those in C_2 disagree. In fact, we have seen such an example in Figure 5.5(a), where the causal effect $\Pr_x(y)$ is unidentifiable in general but becomes identifiable when the given $\Pr^*(\mathbf{V})$ satisfies the CSIs $(Y \perp\!\!\!\perp C \mid A = a, B)$ and $(Y \perp\!\!\!\perp B \mid A = \bar{a}, C)$. On the other hand, there exist causal effects that are unidentifiable for all $\Pr^*(\mathbf{V}) > 0$. A simple example is $\Pr_x(y)$ in Figure 5.5(b), which we formalize as Proposition 22 in Appendix D.1. Knowing $\Pr^*(\mathbf{V})$ leads to several advantages. First, the positivity constraint over $\Pr^*(\mathbf{V})$ becomes immediate

⁸There is an infinite number of such equivalence classes since a causal graph G induces infinitely many distributions $\Pr(\mathbf{V})$.

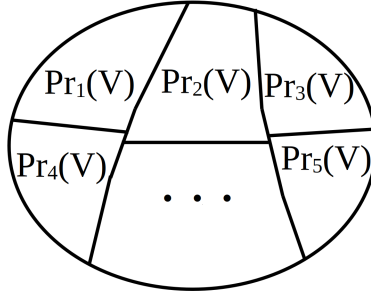


Figure 5.8: Models for $\Pr(\mathbf{V})$

since all the zero entries are determined by $\Pr^*(\mathbf{V})$. Second, all the (well-defined) parameters $\theta_{v|\mathbf{p}}$ over observed variables $V, \mathbf{P}$ can be fixed to constant values in the AC. This allows us to exploit more parameter-specific knowledge such as 0/1 parameters and equal parameters (subsumes CSI) as exhibited by $\Pr^*(\mathbf{V})$. One key observation is that a causal effect $\Pr_{\mathbf{x}}(\mathbf{y})$ is identifiable under a known $\Pr^*(\mathbf{V})$ iff it evaluates to a *fixed constant* under all models that induce $\Pr^*(\mathbf{V})$. This result follows directly from Proposition 6. In fact, the constant can always be evaluated by computing $\Pr_{\mathbf{x}}(\mathbf{y})$ under any single model that induces $\Pr^*(\mathbf{V})$.

Consider the causal graph in Figure 5.9(a) where variable B has states b_1, b_2, b_3, b_4 and all other variables are binary. Suppose $\Pr^*(\mathbf{V}) > 0$ satisfies the conditional probabilities in Figure 5.9(b). We show that the causal effect $\Pr_x(y)$ is identifiable under the fixed $\Pr^*(\mathbf{V}) > 0$ (this causal effect is not identifiable if $\Pr(\mathbf{V})$ is not fixed to $\Pr^*(\mathbf{V})$). Figure 5.10 depicts the AC for the causal effect after plugging in the known constant values for parameters. If we further apply simplification rules on the AC, we obtain an AC which contains a single leaf with constant 0.28. That is, the causal effect $\Pr_x(y) = 0.28$ for all models that induce $\Pr^*(\mathbf{V})$ and is thus identifiable.

The constraint of known $\Pr(\mathbf{V})$ can be combined with other types of constraints to further improve the identifiability of causal effects. We next show an example that assumes both known $\Pr(\mathbf{V})$ and context-specific independences. Consider the causal graph in Figure 5.9(c), where variable Y has three states y_1, y_2, y_3 and all other variables are binary. Suppose the fixed $\Pr^*(\mathbf{V})$ satisfies the conditional probabilities in Figure 5.9(d) and the CSI

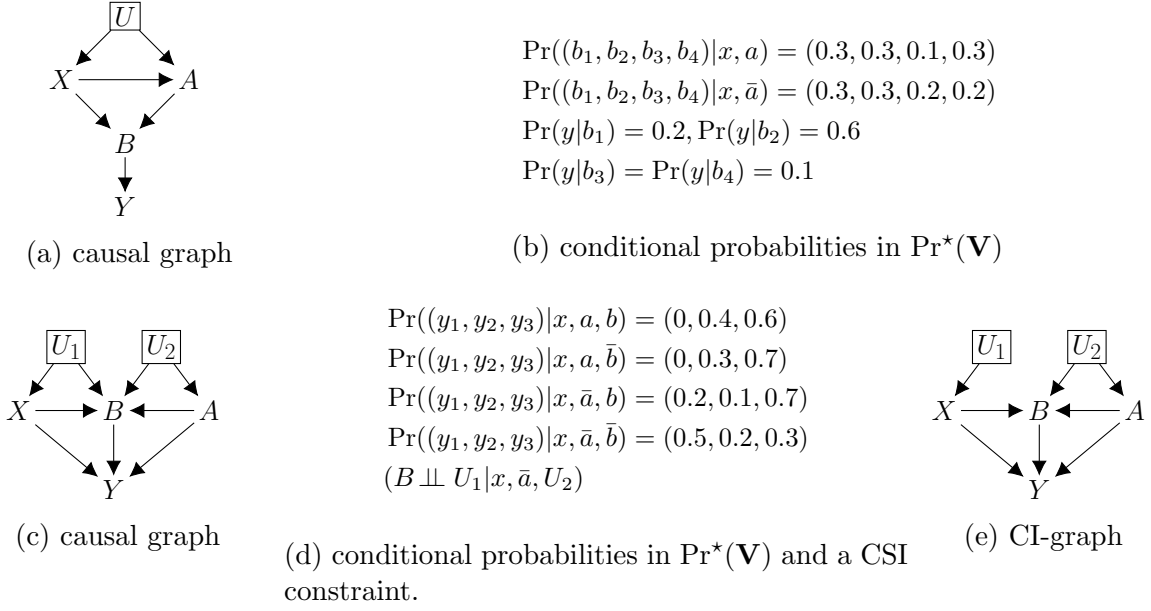


Figure 5.9: Examples for fully-known $\text{Pr}(\mathbf{V})$.

constraint $(B \perp\!\!\!\perp U_1 \mid x, \bar{a}, U_2)$. We next show that the causal effect $\text{Pr}_x(y_1)$ is identifiable under these constraints. Note that strict positivity no longer holds in this example due to the zero entries in the CPT of Y . In fact, the existence of zero entries enables the identifiability since it allows us to prune nodes that evaluate to zero from the AC. Figure 5.10(b) depicts the AC for $\text{Pr}_x(y_1)$ after applying the simplification rules along with the projections and invariant-cut. Similar to the example in Figure 5.7(a), we adopt ID-QFUNC in

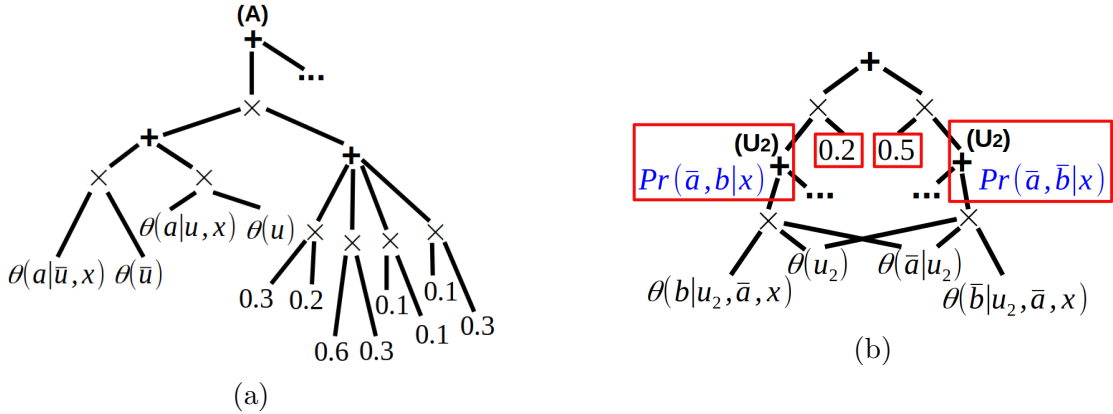


Figure 5.10: ACs constructed for Figure 5.9(a) and Figure 5.9(c) under known $\text{Pr}^*(\mathbf{V})$.

Algorithm 1 to identify the projections. Specifically, the two $+-$ nodes on the invariant-cut requires finding a projection for the c-factor $Q[U_2AB]$ under the context $\bar{a}$, which is feasible since the c-component $\{U_2, A, B\}$ can be obtained from the CI-graph in Figure 5.9(e). The identifying formula in this case is $\Pr_x(y_1) = 0.2 \Pr(\bar{a}, b|x) + 0.5 \Pr(\bar{a}, \bar{b}|x)$, which can be evaluated using the known $\Pr^*(\mathbf{V})$. This example illustrates how the AC-based approach provides a systematic way to combine different types of parametric constraints to further improve identifiability. In general, if the values of parameters involving hidden variables are known, they can be incorporated using the same approach by substituting their constant values and testing identifiability on the resulting simplified AC.

CHAPTER 6

Accelerating Computations with Functional Dependencies

We now consider a different setup where the causal models (CBNs) are fully specified. That is, both the causal graph and all parameters (CPTs) are fully known. Under this setup, all interventional queries (such as causal effects) are identifiable, and causal inference becomes a computational problem that can be reduced to marginal inference. One challenge, however, is that computing marginals on (causal) Bayesian networks is $\#P$ -hard [Coo90, KC15], and, hence, a major research direction has been to leverage additional information to improve the efficiency of marginal inference in practice. One prominent example is the use of *knowledge compilation* technique to exploit local structures such as equal parameters [Dar02, CD07].

The goal of this chapter is to present a formal method for accelerating marginal inference by leveraging the knowledge of qualitative (unknown) functional dependencies. We start in Section 6.1 with background on factor operations and the standard jointree algorithm for computing marginal queries. In Section 6.2, we present a procedure for incorporating qualitative functional dependencies based on a refined notion called *thinned jointrees*. We show that, by using thinned jointrees to compute marginals, we can achieve significant speedups both theoretically and empirically. In particular, we characterize the computational complexity of this procedure in Section 6.3, leading to a tighter complexity bound based on the notion of *causal treewidth*. We then present experimental results in Section 6.4 to further demonstrate the improvements in computational efficiency. The results in this chapter appeared in [CD22, CHD24].

X	Y	f_1	Y	Z	f_2	X	Y	Z	$g = f_1 \cdot f_2$	X	Z	$h = \sum_Y g$
x_1	y_1	0.9	y_1	z_1	2.0	x_1	y_1	z_1	1.80	x_1	z_1	1.85
x_1	y_2	0.1	y_1	z_2	1.8	x_1	y_1	z_2	1.62	x_1	z_2	2.14
x_2	y_1	0.3	y_2	z_1	0.5	x_1	y_2	z_1	0.05	x_2	z_1	0.95
x_2	y_2	0.7	y_2	z_2	5.2	x_1	y_2	z_2	0.52	x_2	z_2	4.18
						x_2	y_1	z_1	0.60			
						x_2	y_1	z_2	0.54			
						x_2	y_2	z_1	0.35			
						x_2	y_2	z_2	3.64			

Table 6.1: Illustrating factors and their key operations, multiplication and sum-out.

6.1 Computing Marginals

6.1.1 Factor Operations

A factor f over a variable set $\mathbf{X} = \{X_1, \dots, X_n\}$, denoted $f(\mathbf{X})$, maps each variable instantiation $\mathbf{x}$ into a non-negative value; see Table 6.1. As discussed in Chapter 2.1, conditional probability tables (CPTs) in (causal) Bayesian networks can be viewed as special types of factors that specify probability distributions. Two main operations performed on factors are called multiplication and sum-out, which we define next.

Multiplying factors $f_1(\mathbf{X})$ and $f_2(\mathbf{Y})$, denoted $f_1 \cdot f_2$, produces a factor $g(\mathbf{Z})$ where $\mathbf{Z} = \mathbf{X} \cup \mathbf{Y}$ and $g(\mathbf{z}) = f_1(\mathbf{x})f_2(\mathbf{y})$ if instantiation $\mathbf{z}$ is consistent with $\mathbf{x}, \mathbf{y}$. For example, in Table 6.1, multiplying factors $f_1(X, Y)$ and $f_2(Y, Z)$ yields the factor $g(X, Y, Z)$, where $g(x, y, z) = f_1(x, y)f_2(y, z)$ for each instantiation x, y, z of variables X, Y, Z .

Summing out variables $\mathbf{W}$ from a factor $g(\mathbf{X})$, denoted $\sum_{\mathbf{W}} g$, produces a factor $h(\mathbf{Z})$ where $\mathbf{Z} = \mathbf{X} \setminus \mathbf{W}$ and $h(\mathbf{z}) = \sum_{\mathbf{w}} g(\mathbf{w}, \mathbf{z})$. For example, in Table 6.1, summing out variable Y from factor $g(X, Y, Z)$ yields the factor $h(X, Z)$ where $h(x, z) = \sum_y g(x, y, z)$ for each instantiation x, z of variables X, Z . The projection operation $\sum_{\mathbf{W}} g(\mathbf{X})$ corresponds to the sum-out operation $\sum_{\mathbf{X} \setminus \mathbf{W}} g(\mathbf{X})$. For example, summing out variable Y from factor $g(X, Y, Z)$ as we did above can be expressed as projecting this factor on variables X, Z : $h(X, Z) = \sum_{X, Z} g(X, Y, Z)$.

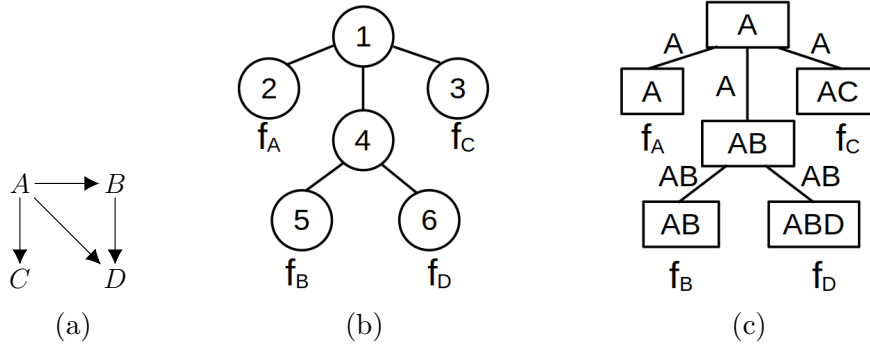


Figure 6.1: A BN, a jointtree, and the jointtree with its separators and clusters.

6.1.2 Jointtrees and Treewidth

The jointtree algorithm is one of the most popular inference methods for computing marginals in Bayesian networks (BNs) [LS88].¹ Given a Bayesian network with factors (CPTs) $\mathcal{F}$, a jointtree consists of a tree structure and a mapping from each factor $f \in \mathcal{F}$ to some node i in the tree. We call node i the *host* of factor f and use $\mathcal{F}_i$ to denote the factors hosted by i . Let f_X denotes the CPT for variable X . Figure 6.1(b) depicts a jointtree for the BN in Figure 6.1(a), where the hosts for CPTs f_A, f_B, f_C, f_D are nodes 2, 5, 3, 6, respectively. As far as inference complexity is concerned, the quality of a jointtree is measured by its width, which is determined by the sizes of its clusters and separators, which we define next.

For each jointtree edge (i, j) , the *separator* $\mathbf{S}_{ij}$ is defined as the set of variables shared between factors on both sides of the edge (i, j) . Consider the edge $(1, 4)$ in Figure 6.1(b). Variables $\{A, C\}$ appear in factors on the side of node 1, and variables $\{A, B, D\}$ appear in factors on the side of node 4. Hence, the separator for the edge is $\{A\} = \{A, C\} \cap \{A, B, D\}$. The separators for the jointtree are marked near the edges in Figure 6.1(c). The clusters of a jointtree are defined as follows. For each leaf node i (one with a single neighbor), the cluster $\mathbf{C}_i$ is defined as the variables of factors hosted at i ; for each non-leaf node i , $\mathbf{C}_i$ is the union of separators adjacent to i . For example, in Figure 6.1(b), the cluster of leaf node 5 is $\{A, B\}$

¹Since causal Bayesian networks (CBNs) are a subtype of Bayesian networks whose graphs specify causal relationships, the results developed here for Bayesian networks also apply to CBNs.

since this node hosts the only factor $f_B(A, B)$. Moreover, the cluster for non-leaf node 4 is $\{A, B\}$, which is the union of its adjacent separators $\{A\}, \{A, B\}, \{A, B\}$. The clusters for the jointree are shown inside nodes in Figure 6.1(c).

The *width* of a jointree is defined as the largest cluster size minus one, and the *treewidth* of a Bayesian network is defined as the least width attainable by any jointree for the network [RS86]. The width of the jointree in Figure 6.1(c) is 2. The treewidth of the BN in Figure 6.1(a) is also 2 as we cannot obtain a jointree with a width lower than 2. Hence, the jointree in Figure 6.1(c) is optimal. To simplify some of the upcoming formulations, we will assume that CPTs are hosted only by leaf nodes of a jointree (recall that a leaf node has only one neighbor). We can convert any jointree to one that satisfies this property without changing the jointree width as follows. For each non-leaf node i that hosts a CPT f , we add a new leaf node j as a neighbor of i and make j the host of CPT f ; see [Dar09, Ch. 7] for the soundness of this technique.

The notions of width and treewidth are crucial as they dominate the complexity of the jointree algorithm. Specifically, the time/space complexity for the jointree algorithm is $O(n \cdot \exp(w))$, where n is the number of nodes in the jointree and w is the width. Hence, one normally seeks a jointree with small width to reduce the computations required by the jointree algorithm, albeit finding a jointree with an optimal width is known to be NP-hard [AP89]. Jointrees can be used to compute marginals through the following message-passing algorithm that employs factor operations (multiplication and summation); see [Dar09, Ch. 7] for details.

Theorem 9. *Consider a jointree for factors $f_1, \dots, f_n$. Define the message from jointree node i to its neighbor j as:*

$$\mathcal{M}_{ij} = \begin{cases} \sum_{\mathbf{s}_{ij}} \mathcal{F}_i & \text{for leaf node } i \\ \sum_{\mathbf{s}_{ij}} \prod_{k \neq j} \mathcal{M}_{ki} & \text{for non-leaf node } i \end{cases}$$

For all jointree edges (i, j) , $\mathcal{M}_{ij}\mathcal{M}_{ji} = \sum_{\mathbf{s}_{ij}} f_1 \cdots f_n$.

The message passing algorithm allows us to compute the marginal over any separator $\mathbf{S}_{ij}$.² To illustrate, consider the separator $\{A, B\}$ for edge $(4, 5)$. The marginal over AB can be computed using the message passing algorithm as $\Pr(AB) = \mathcal{M}_{45} \cdot \mathcal{M}_{54}$, where

$$\mathcal{M}_{45} = \sum_{AB} [\sum_A [\sum_A f_A] [\sum_A f_C]] [\sum_{AB} f_D], \quad \mathcal{M}_{54} = \sum_{AB} f_B$$

6.2 Exploiting Qualitative Functional Dependencies

When the knowledge of qualitative functional dependencies is available, we can accelerate the computation of marginals and break the treewidth barrier. The *qualitative*, rather than quantitative, nature of these functional dependencies allows speedups to be made independent of the model parameters, making the approach applicable to more general scenarios where the parameters (functions) are unknown (e.g., in learning tasks) or dynamically changing (e.g., in domains such as the stock market).

The improvement relies on a new data structure called *thinned jointtrees*, which are obtained from classical jointtrees by removing variables from separators/clusters while adding additional nodes to the jointtree. Consider a jointtree with n nodes and width w . Computing marginals using this jointtree will have a time/space complexity of $O(n \cdot \exp(w))$. By thinning the jointtree, we obtain a thinned jointtree of size $n' \geq n$ and width $w' \leq w$, which has complexity $O(n' \cdot \exp(w'))$. This can lead to substantial speedups if w' is much smaller than w . To illustrate, Figure 6.2 depicts an optimal, classical jointtree with width 3 and a thinned jointtree with width 2 for the same BN structure. The use of thinned jointtrees relies on key variable elimination results introduced in [Dar20], which we review in Appendix E.1.

²By construction, every non-leaf node in the Bayesian network appears in some separator. To ensure that every variable appears in some separator, add an auxiliary child C_i for each leaf X_i in G with factor $g_i(C_i, X_i)$ such that $g_i(c_i, x_i) = 1$.

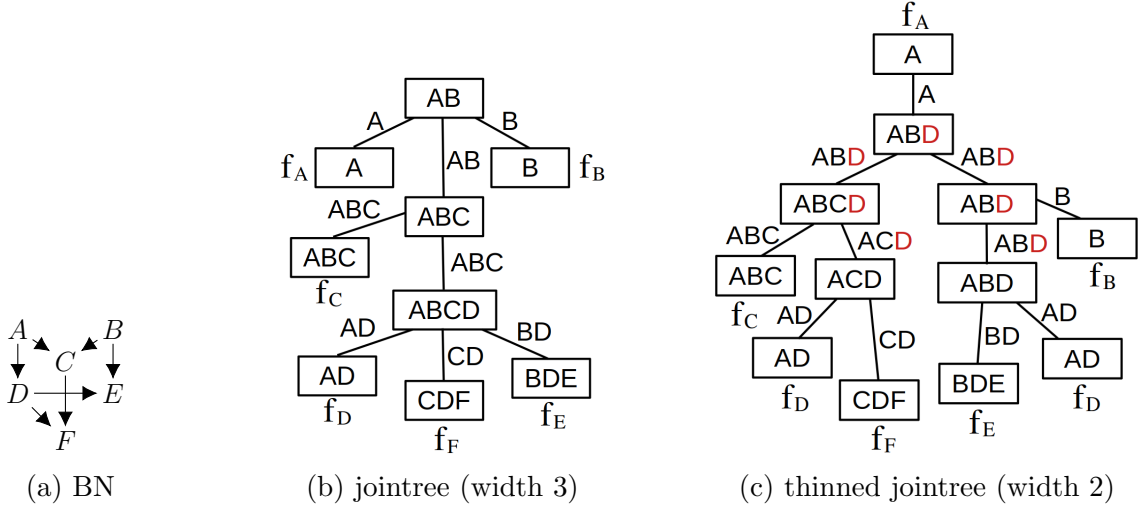


Figure 6.2: A CPT f appears next to a jointree node i iff f is hosted by i . The mechanism for the functional variable D is replicated in (c), where red variables are thinned.

The goal of this section is to present the overall procedure for constructing thinned jointrees; detailed properties will be discussed in Section 6.3. At a high level, the procedure involves three steps (ingredients): (1) mechanism replication, (2) jointree construction, and (3) jointree thinning. These steps rely on the notions of mechanisms and feeding factors, defined as follows. The CPT for a variable X , denoted $f(X, \mathbf{P})$, is called a *mechanism* for X (or *X -mechanism*) if X is functional, and is called *Y -feeding* if $Y \in \mathbf{P}$.

Step 1: Mechanism Replication

Given the CPTs $\mathcal{F}$ of a BN, we first create a *replication* $\mathcal{F}' \supseteq \mathcal{F}$ by adding copies of the mechanisms to $\mathcal{F}$. This is justified by Theorem 20 in Appendix E.1, which states that the product of CPTs in $\mathcal{F}$ is identical to the product of CPTs in its replication $\mathcal{F}'$. Consider the BN in Figure 6.3(a) which has five CPTs $\{f_A, f_B, f_C, f_D, f_E\}$ and suppose variables B, C are functional; that is, their CPTs f_B, f_C correspond to mechanisms. Instead of constructing a jointree for the original CPT set, we can construct a jointree for the replication $\{f_A, f_B, f_B, f_B, f_C, f_C, f_D, f_E\}$ which has three copies of CPT f_B and two copies of CPT f_C . Again, since replicating mechanisms preserves the joint probability over all variables,

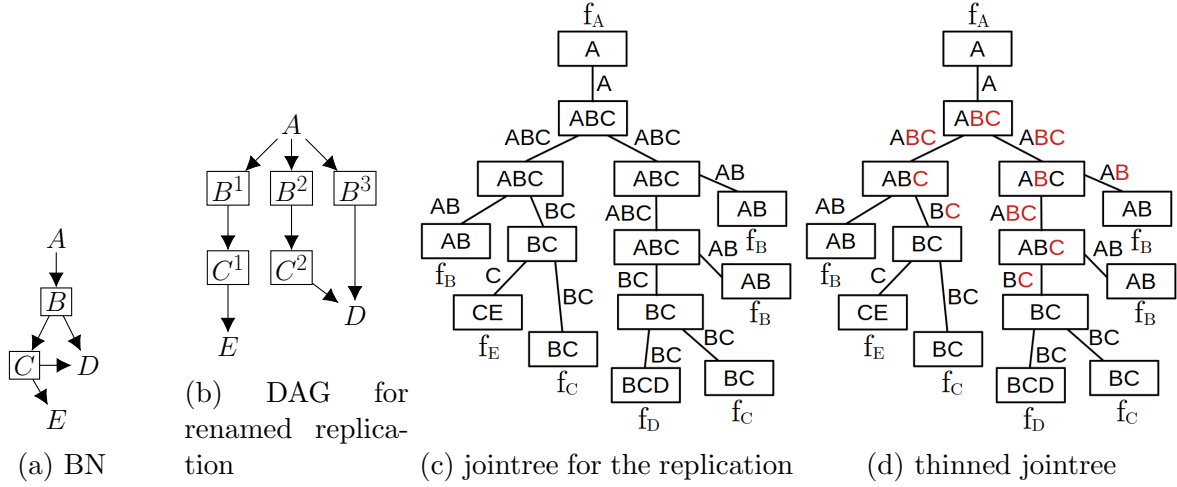


Figure 6.3: A Bayesian network, its renamed replication, a jointtree and a thinned jointtree for the replication. Variables B and C in the BN are functional. Red variables are thinned.

i.e., $\Pr(ABCDE) = f_A f_B f_C f_D f_E = f_A f_B f_B f_B f_C f_C f_D f_E$, we can use the jointtree for the replication to compute marginals. The process of mechanism replication generally leads to jointtrees with more nodes but the replicated mechanisms will allow us to remove functional variables from clusters, leading to thinned jointtrees with smaller widths.

One particular replication strategy of interest is called the *complete replication* strategy [Dar20]. The complete replication $\mathcal{F}'$ for $\mathcal{F}$ is obtained by replicating mechanisms in $\mathcal{F}$ such that for each functional variable X , (i) if $\mathcal{F}'$ contains no X -feeding factor, then it contains exactly one X -mechanism (ii) otherwise, $\mathcal{F}'$ contains an equal number of X -mechanisms and X -feeding factors. Algorithm 3 shows a procedure for generating the complete replication. The complete replication always exists and is unique. It is “complete”, or optimal, in the sense that adding more mechanisms to the replication cannot lead to more speedups. A drawback of this type of replication, however, is that its size may grow exponentially in the worst case; see Section 6.3 for a detailed discussion and possible remedies.

Algorithm 3 Complete Replication

```
1: procedure REPLICATE(Bayesian network  $M$ , Functional nodes  $\Gamma$  in  $M$ )
2:    $\Sigma \leftarrow$  multi-set of CPTs in  $M$ 
3:   for each node  $X$  in  $\Gamma$  (bottom-up traversal) do
4:     if  $X$  is a leaf then continue
5:      $n \leftarrow$  number of  $X$ -feeding factors in  $\Sigma$ 
6:      $\Sigma \leftarrow \Sigma \cup \{n - 1 \text{ copies of the CPT for } X\}$ 
7:   return  $\Sigma$ 
```

Step 2: Jointree Construction

After obtaining a replication $\mathcal{F}'$ that contains replicated mechanisms, our next step is to construct a jointree for $\mathcal{F}'$ that can lead to potential jointree thinnings. While there are various ways to construct the jointree, we present an approach that targets complete replications.

We first rename the functional nodes in $\mathcal{F}'$. Specifically, we go over each X -mechanism in $\mathcal{F}'$ and rename X in the i^{th} X -mechanism to X^i . Similarly, we go over each X -feeding factors and rename X in the i^{th} X -feeding factor to X^i .³ We call the renamed functional nodes *functional replicas* for X , or simply X -replicas. The renaming process allows us to treat functional replicas as distinct nodes. For example, after renaming all functional nodes in the complete replication for the BN in Figure 6.3(a), we obtain $\{f_A(A), f_{B^1}(AB^1), f_{B^2}(AB^2), f_{B^3}(AB^3), f_{C^1}(B^1C^1), f_{C^2}(B^2C^2), f_D(B^3C^1D), f_E(C^2E)\}$. The renamed replication can also be visualized by the DAG shown in Figure 6.3(b). We then construct a jointree for the renamed replication using standard methods discussed in [Dar09, Ch 9] and replace functional nodes with their original names in all separators and clusters. Figure 6.3(c) shows a jointree constructed for the complete replication $\mathcal{F}'$.

³For replications that are not complete, we may rename X to any X^j appearing in one of the X -mechanisms.

Step 3: Jointree Thinning

After constructing a jointree for a replication $\mathcal{F}'$, we reduce the width of the jointree by thinning (removing) functional variables from separators and clusters based on some thinning rules. While we will formally define thinning rules in the next section, we provide a brief summary here. The first rule removes a functional variable X from a separator $\mathbf{S}_{ij}$ if edge (i, j) is on the path between two leaf nodes that host f_X and every separator on that path contains X . The second rule removes variable X from a separator $\mathbf{S}_{ij}$ if no other separator $\mathbf{S}_{ik}$ (where k is a neighbor of i) contains X , or no other separator $\mathbf{S}_{kj}$ (where k is a neighbor of j) contains X . For instance, applying thinning rules to the jointree in Figure 6.3(c) yields the thinned jointree in Figure 6.3(d). We can then use the thinned jointree in place of the classical jointree to compute marginals more efficiently. Consider the thinned jointree in Figure 6.2(c) and the separator adjacent to the top-most node containing A . We can compute the marginal $\Pr(A)$ as follows:

$$\begin{aligned} \Pr(A) = & [\sum_A^{\perp} f_A] [\sum_A^{\perp} [\sum_{AB}^{\perp} [\sum_{AC}^{\perp} [\sum_{CD}^{\perp} f_F] [\sum_{AD}^{\perp} f_D]] [\sum_{ABC}^{\perp} f_C]] \\ & \cdot [\sum_{AB}^{\perp} [\sum_{AB}^{\perp} [\sum_{AD}^{\perp} f_D] [\sum_{BD}^{\perp} f_E]] [\sum_B^{\perp} f_B]] \end{aligned}$$

where the projected sets correspond exactly to the separators in the thinned jointree.

6.3 Thinned Jointrees: Definition and Properties

6.3.1 Jointree Thinning

We formally define the notion of *jointree thinning*, which leads to the definition of thinned jointrees. Given a classical jointree constructed for some replication $\mathcal{F}'$, jointree thinning assigns a new set of separators for the jointree by removing functional variables. Note that the choice of replication $\mathcal{F}'$ does not need to be optimal here; constructing optimal replications will be discussed in the next subsection.

We start with the definition of X -connections, which will be used in the definition of

jointree thinning.

Definition 13. A jointree node i is said to be X -connected to a factor f iff i hosts f or X appears in every separator on the path between i and some leaf node j that hosts factor f .

Definition 14. A jointree thinning maps every edge (i, j) in the jointree to a set of variables $\mathbf{S}_{ij}^* \subseteq \mathbf{S}_{ij}$, called a thinned separator, and satisfies two properties. First, for each functional variable $X \in \mathbf{S}_{ij}^*$, we have:

- (a) Node i is not X -connected to any X -mechanism on the i -side of edge (i, j) , or node j is not X -connected to any X -mechanism on the j -side of the edge.
- (b) If node i is not a leaf, then $X \in \mathbf{S}_{ik}^*$ for some $k \neq j$.
- (c) If node j is not a leaf, then $X \in \mathbf{S}_{jk}^*$ for some $k \neq i$.

Second, no other mapping from edges to supersets of $\mathbf{S}_{ij}^*$ satisfies the above property.

The separators $\mathbf{S}_{ij}$ of a jointree are determined by the locations of factors (the leaf nodes they are hosted at). Hence, the separators of a jointree are unique. However, a thinned separator $\mathbf{S}_{ij}^*$ depends on both the locations of factors and other thinned separators. Hence, a jointree may have multiple thinnings. For example, consider the Bayesian network in Figure 6.4 with two functional variables $\{B, C\}$ and nine factors $\mathcal{F} = f_A, f_B, \dots, f_I$. Consider now the replication $\mathcal{F}' \supseteq \mathcal{F}$ which results from duplicating mechanisms f_B and f_C once (that is, $\mathcal{F}'$ has 11 factors). Figure 6.5 depicts a jointree for the replication $\mathcal{F}'$ and two different thinnings obtained using Definition 14.

To evaluate the optimality of jointree thinnings, we assign clusters to the jointree nodes in the standard manner, using thinned separators in place of classical ones. The resulting jointree, with thinned separators and clusters, is called a *thinned jointree*. Its width can then be defined in the standard way based on the sizes of its clusters.

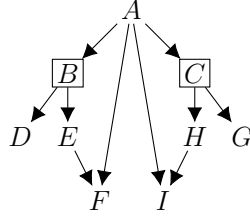


Figure 6.4: A Bayesian network with functional variables B and C .

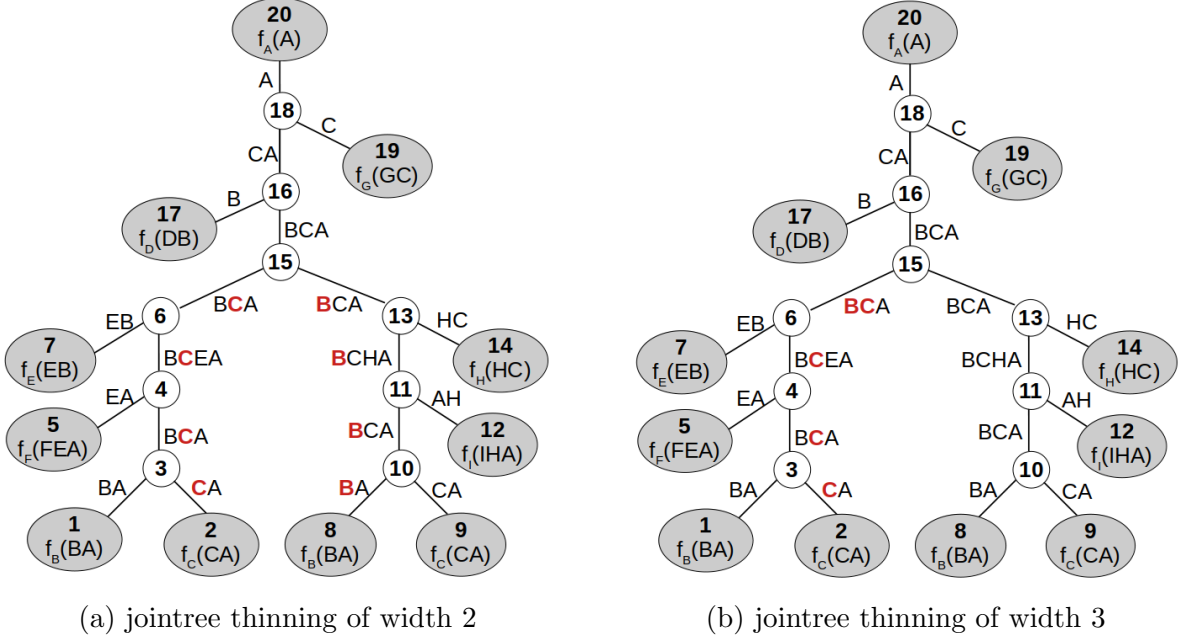


Figure 6.5: Two jointree thinnings. Each edge (i, j) is marked by separator S_{ij} . Red variables are not in the thinned separator S_{ij}^* .

Definition 15. A *thinned jointree* is a jointree in which edges are annotated with thinned separators and nodes are annotated with thinned clusters.⁴ The *width* of a thinned jointree is the size of its largest thinned cluster minus one.

The thinned jointree in Figure 6.5(a) has width 2, and the one in Figure 6.5(b) has width 3. Since the complexity of jointree algorithm is exponential in width, using the first thinned jointree results in faster marginal computations.

It remains to show that thinned jointrees as given by Definition 15 can indeed be used

⁴Thinned jointrees were also called “causal jointrees” in [CD22].

to compute marginals. We establish this by showing that they preserve the correctness of the message passing algorithm.

Theorem 10. *Theorem 9 continues to hold if we use thinned separators $\mathbf{S}_{ij}^*$ (as given by Definition 14) instead of classical separators $\mathbf{S}_{ij}$.*

Theorem 21 in Appendix E.1 further suggests the optimality of jointree thinnings. Definition 14 tells us what a thinning is but it does not tell us how to obtain one. We next provide a set of *thinning rules* that will generate every thinning admitted by Definition 14.

Theorem 11. *The separators $\mathbf{S}_{ij}^*$ constitute a valid jointree thinning iff it can be obtained by starting with $\mathbf{S}_{ij}^* = \mathbf{S}_{ij}$ and then removing variables from $\mathbf{S}_{ij}^*$ according to the following rules, until no rules can be applied. Remove functional variable X from $\mathbf{S}_{ij}^*$ if either*

- (a) *Node i is X -connected to some X -mechanism on the i -side of edge (i, j) and node j is X -connected to some X -mechanism on the j -side of the edge; or*
- (b) *$X \notin \mathbf{S}_{ki}^*$ for all $k \neq j$ when node i is not a leaf; or*
- (c) *$X \notin \mathbf{S}_{jk}^*$ for all $k \neq i$ when node j is not a leaf.*

That is, every thinned jointree can be obtained by applying the two thinning rules to exhaustion. We will use $R_a(i, j, X)$ to denote that Rules (a) is applicable to variable X and edge (i, j) , and refer to each as a *rule application*. Similarly for Rules (b) and (c). A jointree thinning can now be specified using a sequence of rule applications. The thinning in Figure 6.5(a) corresponds to $R_a(6, 15, C)$, $R_c(4, 6, C)$, $R_c(3, 4, C)$, $R_c(2, 3, C)$, $R_a(13, 15, B)$, $R_c(11, 13, B)$, $R_c(10, 11, B)$, $R_c(8, 10, B)$. The one in Figure 6.5(b) corresponds to $R_a(6, 15, C)$, $R_c(4, 6, C)$, $R_c(3, 4, C)$, $R_c(2, 3, C)$, $R_c(6, 15, B)$.

Thinning rules mechanize the construction of thinned jointrees, but finding an optimal sequence of rule applications remains a computationally challenging task given the large number of such sequences. Hence, one needs either sophisticated search algorithms or a

heuristic to decide which thinning rule to apply and when. One heuristic that we found effective is to prefer $R_a(i, j, X)$ with the largest $\mathbf{S}_{ij}^*$, followed by X that is contained in the fewest neighboring separators, followed by minimizing the number of X -connected X -mechanisms on either side of edge (i, j) . We apply Rules (b) and (c) after exhausting all applications of Rule (a).

6.3.2 Mechanism Replication

The definition of thinning developed in the previous subsection was with respect to a particular replication $\mathcal{F}$ and a particular jointree for $\mathcal{F}$. Some replications are better than others in that they lead to thinned jointrees of smaller width. We formalize this next.

Definition 16. *The width of a replication $\mathcal{F}$ is defined as the minimum width attained by any thinned jointree for $\mathcal{F}$.*

Given a replication $\mathcal{F}$, we need to examine two search spaces before we can determine its width. First, we must choose a jointree for the factors in $\mathcal{F}$. Second, we must choose a particular thinning of the jointree. Hence, determining the width of a replication is not a straightforward task. Moreover, the width of a replication is not the only measure of its quality, as we also need to consider its size (i.e., the number of factors it contains). Consider two replications $\mathcal{F}_1$ and $\mathcal{F}_2$ with respective sizes n_1 and n_2 . Suppose now that replication $\mathcal{F}_1$ has width w_1 and replication $\mathcal{F}_2$ has width w_2 . This means that there exists a thinned jointree for replication $\mathcal{F}_1$ of width w_1 and no other thinned jointree can have a smaller width (and similarly for replication $\mathcal{F}_2$). If we use these optimal thinned jointrees, inference using these replications can be done in $O(n_1 \cdot \exp(w_1))$ and $O(n_2 \cdot \exp(w_2))$ time and space, respectively. One may be tempted to choose the replication with smaller width since complexity is exponential in width but linear in size. However, the size of a replication may also be exponential as we show next. In fact, the key result of this section is that the complete replication strategy mentioned earlier in Section 6.2, shown in Algorithm 3, satisfies

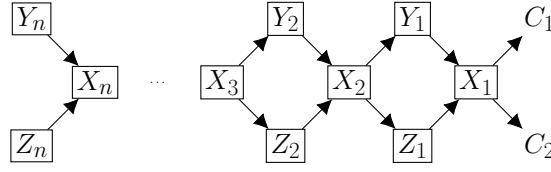


Figure 6.6: A BN with an exponentially sized complete replication.

two interesting properties. First, it is optimal in width: no other replication strategy will have a smaller width. Second, it can lead to replications of exponential size.

We start with the optimality of complete replications.

Theorem 12. *Let $\mathcal{F}$ and $\mathcal{F}'$ be two replications for a Bayesian network. If replication $\mathcal{F}$ is complete, then its width is no greater than the width of replication $\mathcal{F}'$.*

That is, we cannot achieve smaller widths than those achieved by complete replication regardless of how many mechanisms we replicate. To see why the size of complete replications can be exponential, consider the family of BNs in Figure 6.6, which contains $3n + 2$ nodes. Algorithm 3, which generates complete replications, visits the functional nodes bottom-up and creates 2^i mechanisms for each X_i .

We next provide a bound on the number of factors in a complete replication which suggests a method for controlling the potential blow up in its size.

Proposition 8. *Consider a Bayesian network with n nodes. Let c be the largest number of children for any node and let k be the length of the longest functional chain (a path in which all nodes are functional). Then the complete replication will contain at most nc^k factors.*

This bound immediately provides a method for controlling the number of replicas in a complete replication. If we treat variable $X_{n/2}$ in Figure 6.6 as a non-functional variable, the length of the largest functional chain will be cut by half. Hence, by selectively ignoring some functional variables we can bound the size of functional chains and therefore ensure that Algorithm 3 will produce replications with size polynomial in the number of variables.

We are now ready to define the notion of causal treewidth, which captures the optimal width attainable by any polynomial-size replication.

Definition 17. *Consider a DAG G with n nodes, some of which are declared as functional. The causal treewidth of DAG G is the smallest width attained by any replication $\mathcal{F}$ for G where the size of $\mathcal{F}$ is polynomial in n .*

Causal treewidth is guaranteed to be no greater than classical treewidth, and hence provides a tighter complexity bound for marginal inference under functional dependencies. In general, the optimal complexity for computing marginals using thinned jointress is no worse than $O(n \cdot \exp(w))$, where n is the size of the (optimal) replication and w is the causal treewidth. In cases where many variables are functional, the causal treewidth may be significantly smaller than the classical treewidth. For example, [Dar21a] identified a family of BNs with $O(n^2)$ nodes, bounded depth and treewidth $n + 1$, while constructing thinned jointrees of width 2 for the family. However, determining the causal treewidth can be challenging, as it requires searching for optimal replications and thinned jointrees.

For the class of BNs with bounded functional chains, one can use the complete replication $\mathcal{F}$ to determine the causal treewidth. That is, determining causal treewidth becomes a matter of finding an optimal thinned jointree for the factors in $\mathcal{F}$. The situation is more intricate for BNs with unbounded functional chains. The complete replication cannot be used in this case and one must search among replications of polynomial size. It remains an open question whether the space of replications to be explored can be restricted to subsets of the complete replication as suggested earlier.

6.4 Experiments

We present two sets of experiments. The first set compares the widths of (thinned) jointrees constructed using different methods, which demonstrates the effectiveness of the thinning rules in Theorem 11. The second set highlights the impact of replication strategies and how

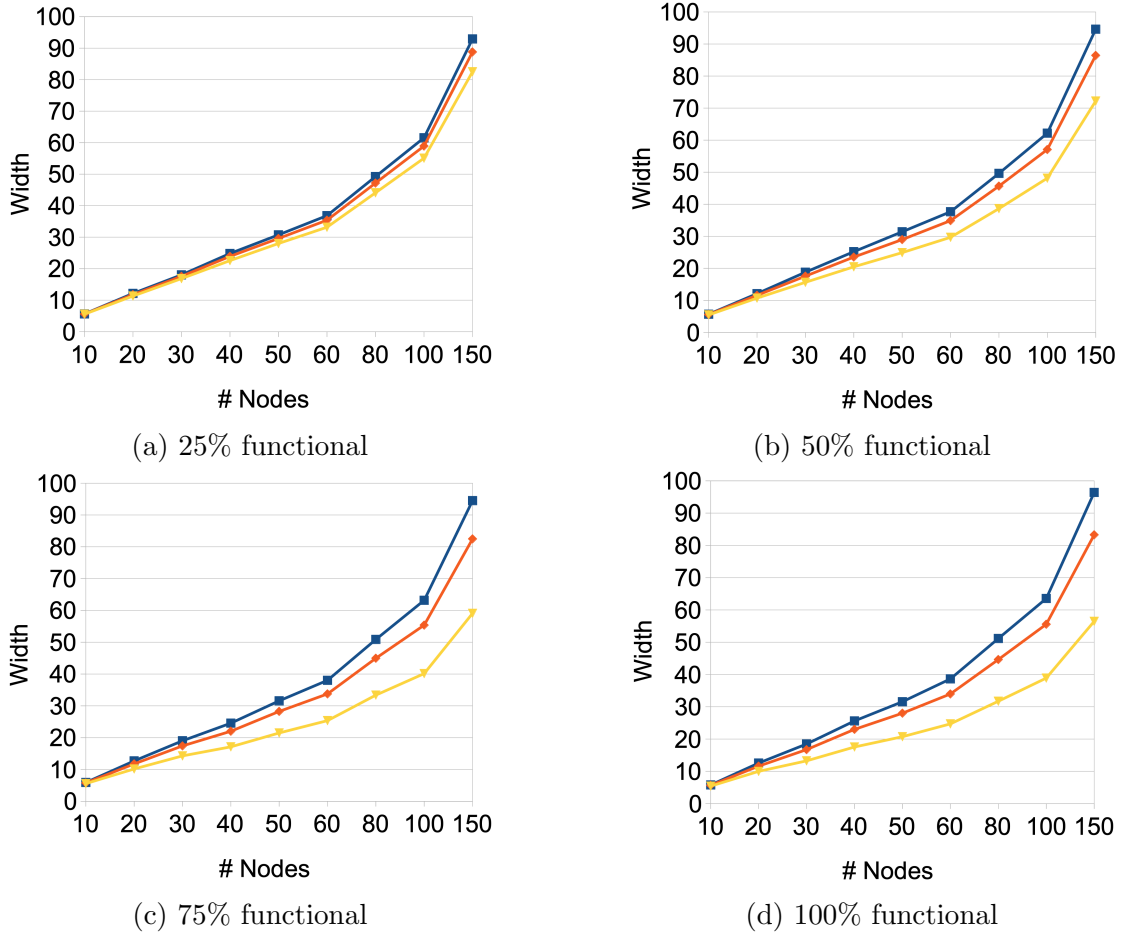


Figure 6.7: Comparing the thinned jointrees constructed using Theorem 11 with the ones in [Dar20]. Functional variables are restricted to be non-root.

these strategies interact with jointree construction methods.

Figure 6.7 compares the mean widths of thinned jointrees constructed using the rules in previous work [Dar21b] and the rules introduced in Theorem 11 over 10 random jointrees, for the factors of complete replications generated by Algorithm 3. The plots vary the number of Bayesian network nodes from 10 to 150 and consider different percentages of functional nodes (25, 50, 75 and 100) which are restricted to be non-root nodes.⁵ Specifically, the plots show

⁵Our method for generating a Bayesian network with nodes $X_1, \dots, X_n$ assumes that each node has at most five parents. We visit nodes X_i from $i = 1$ to $i = n$. When visiting node X_i , we randomly choose a number from $\{0, \dots, \min(5, i-1)\}$ to represent the number of parents for X_i and then randomly choose that many parents from $X_1, \dots, X_{i-1}$.

func%	replication type							
	c	$5c$	$10c$	$15c$	$20c$	$25c$	$30c$	complete
25%	126.2	334.1	594.6	854.3	1114.1	1373.6	1634.2	181.1
50%	151.2	561.4	1076.0	1588.9	2104.1	2618.3	3132.3	513.4
75%	178.7	799.3	1575.6	2352.9	3128.2	3907.0	4682.3	2564.3
100%	187.0	867.7	1721.0	2573.5	3429.6	4280.1	5132.8	4448.0

Table 6.2: Average size of replications. Replication nc means: the number of replicas for a node X is between $(n - 1)c$ and nc where c is the number of children for node X .

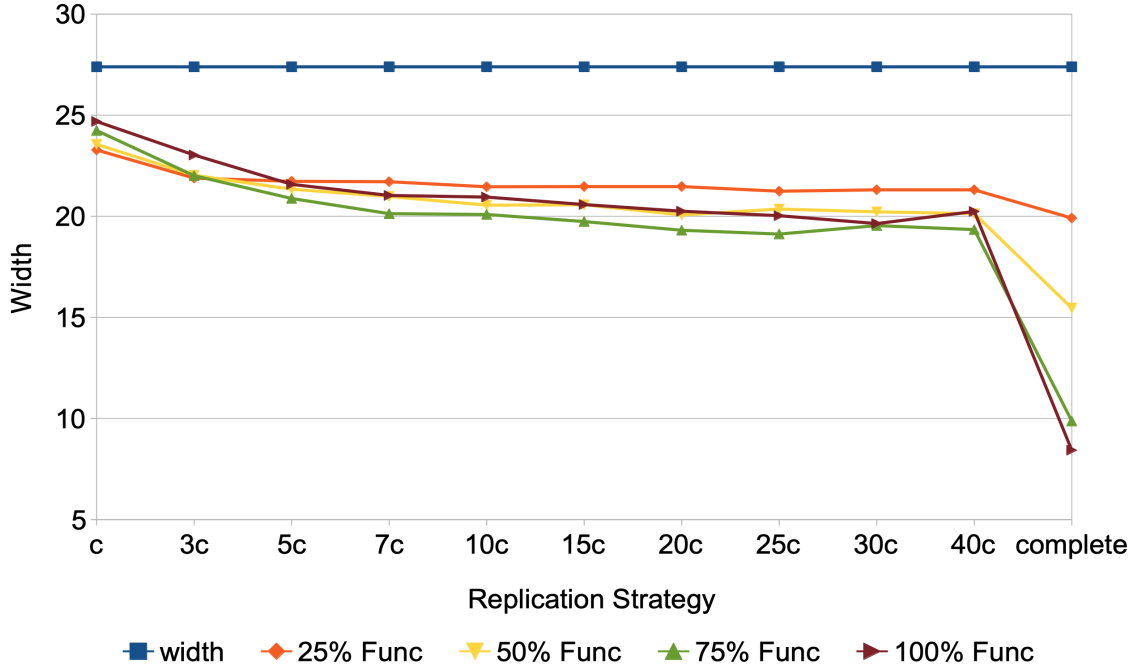


Figure 6.8: Illustrating the impact of replication strategies.

the widths of (1) a classical jointtree (blue), (2) a thinned jointtree obtained by [Dar21b] (red), and (3) a thinned jointtree obtained using the rules in Theorem 11 (yellow). Four patterns are clear: more thinning takes place as the number of functional nodes increases; the proposed thinning rules are much more effective; the gap between the two sets of rules widens as the number of nodes and the proportion of functional nodes increase; the exploitation of functional dependencies can lead to significant reductions in inference complexity.

Our second experiment compares the complete replication strategy with random replications of increasing size, while varying the percentage of functional, non-root nodes in a

Bayesian network (25, 50, 75, 100). Table 6.2 shows the size of random and complete replications, with some random replications being larger than complete replications. The sizes of random replications are determined by a parameter n as follows. For each functional variable X with c children, we create r replicas of the X -mechanism, where r is a random integer in $[(n - 1)c, nc]$.⁶

Figure 6.8 shows the mean width for jointrees and thinned jointrees where each data point is an average over 100 random Bayesian networks, each containing 100 nodes. A few patterns are clear. First, thinned jointrees always produce smaller widths than classical jointrees, and quite substantially smaller, even when using random replications. Second, complete replications consistently yield smaller compared to random replications, particularly when the number of functional nodes is largest (100%). Third, increasing the size of a random replication generally correlates with a decrease in width, but only up to a point; beyond that, further increasing the replication size does not help. These patterns are consistent with our theoretical findings.

⁶When the number of X -feeding factors did not match the number of X -mechanisms, we rename variables in the X -feeding factors distinctly to the extent possible and assign the remaining randomly.

CHAPTER 7

Extensions to Counterfactual Reasoning

We now move beyond causal effects and consider a more sophisticated class of causal queries that involve reasoning about counterfactual events. These counterfactual queries may involve observations or interventions on conflicting states of variables; for example, the probability that a student would have passed the exam (y) had they studied hard (x), given that they did not study ($\bar{x}$) and failed the exam ($\bar{y}$). The query contains conflicting states for variables X and Y , and is commonly denoted as $\Pr(y_x|\bar{x}, \bar{y})$. Such counterfactual queries belong to the third layer of the causal hierarchy and are among the most challenging to answer, as they require reasoning across hypothetical and factual scenarios that cannot be simultaneously realized [pea18].

To compute counterfactual queries, it is standard to consider *structural causal models* (SCMs) [GP98, Hal00], which we will review in Section 7.1. A key assumption of SCMs is that all non-root variables are functional, and therefore the results presented in Chapter 6 on exploiting functional dependencies can be leveraged to accelerate the computation of counterfactual queries as well. To achieve this, we derive a complexity bound for counterfactual reasoning in terms of the causal treewidth, starting with the simpler case for counterfactual queries involving two conflicting worlds (e.g., $\Pr(y_x|\bar{x}, \bar{y})$ in Section 7.2, and extending the results to general queries involving more conflicting worlds (e.g. $\Pr(y_x, \bar{y}_{\bar{x}}|x, \bar{y})$) in Section 7.3. We briefly discuss how these results apply to *partially specified SCMs* (with unknown parameters) in Section 7.4. We demonstrate the effectiveness of our methods empirically in Section 7.5. The results in this chapter appeared in [HCD23, CD22].

the probability that the same applicant would have been admitted ($A = a$) if the college had ignored standardized exam scores in its admission process (for example, by setting $S = s$ for all applicants)? This counterfactual query can be written as $\Pr(a_s|\bar{a}, \bar{s})$, where $\bar{a}, \bar{s}$ represents the evidence in the actual world, and a_s represents the outcome under the intervention in the counterfactual (imaginary) world.

To compute the query, we construct a twin network shown in Figure 7.2(b) using the following procedure. We start with the base network and add *duplicate variables* to it. For each internal variable X , we create a duplicate variable $[X]$. For each parent Y of X , if Y is a root, then we direct an edge from Y to $[X]$; otherwise, we direct an edge from $[Y]$ to $[X]$. For each duplicate node $[X]$, we assign it the same CPT as the original node X . For convenience, we assume that the duplicate node for each root node U is itself, i.e., $[U] = U$, and use $[\mathbf{S}]$ to denote $\{[S] : S \in \mathbf{S}\}$ for any variable set $\mathbf{S}$. Intuitively, the base network represents the events in the actual world, while the duplicates represent the events in the other counterfactual world. The two worlds share the same set of exogenous variables, ensuring that the background factors are consistent across both worlds.

Once a twin network [BP94b] is constructed, we can reduce the counterfactual query to a marginal query so that existing inference methods can be applied. We first incorporate the observations and interventions for each world. In the above college admission example, the counterfactual query $\Pr(a_s|\bar{a}, \bar{s})$ can then be computed as $\Pr([A] = a_{[S]=s} | A = \bar{a}, S = \bar{s})$ on the modified twin network, which becomes an interventional query that no longer involves conflicting states. This can be further reduced to a standard marginal query $\Pr([A] = a | A = \bar{a}, S = \bar{s})$ by asserting evidence ($A = \bar{a}, S = \bar{s}, [A] = a$) and intervening on $[S] = s$.

The key insight is that, since counterfactual reasoning can be reduced to marginal problems on twin networks, its complexity is governed by that of marginal inference on twin networks. It turns out that the inference complexity on twin networks is bounded by the complexity of computing marginals on the base networks, indicating that counterfactual reasoning is no harder than associational/interventional reasoning as long as the SCMs are fully

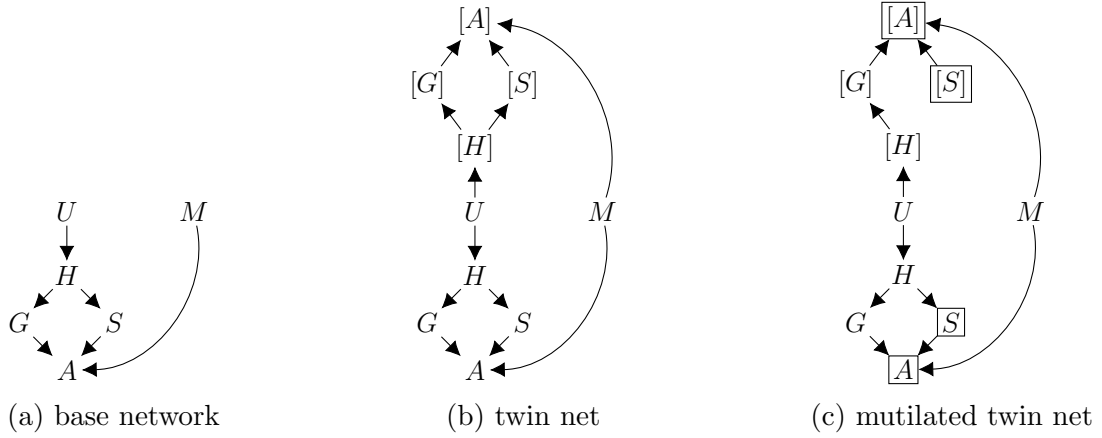


Figure 7.2: Computing counterfactual $\Pr(a_s|\bar{a}, \bar{s})$ using a twin network. Variables with evidence or interventions are indicated with boxes.

specified. This is the focus of the remainder of the chapter.

7.2 Bounding the Causal Treewidth of Twin Networks

Recall from Chapter 6 that in the presence of (qualitative) functional dependencies — which are prevalent in SCMs — the complexity of marginal inference can be further reduced by employing thinned jointrees, yielding a tighter complexity bound called causal treewidth.

Since functional dependencies are abundant in SCMs (all internal nodes are functional), we can apply the same approach based on thinned jointrees to accelerate the computation of counterfactual queries on twin networks. We next provide an even stronger result showing that we can bound the complexity of counterfactual reasoning using the complexity of observational/interventional reasoning. In particular, given a thinned jointree for the base network with width w , which we call a (*thinned*) *base jointree*, we provide a procedure for constructing a thinned jointree for the twin network with width no greater than $2w + 1$, which we call a (*thinned*) *twin jointree*.

The construction of twin jointrees operates on an oriented jointree, obtained from the base jointree $\mathcal{T}$ by directing its edges away from some *root* r that does not host any family

Algorithm 4 Jointree to Twin Jointree

```
1: procedure MAKE-TWIN-JOINTREE( $\mathcal{T}, r, p$ )
2:    $\Sigma \leftarrow$  leaf nodes at or below node  $r$ 
3:   if nodes in  $\Sigma$  only host families for root variables then
4:     return
5:   if nodes in  $\Sigma$  only host families for internal variables then
6:     duplicate the jointree nodes rooted at node  $r$ 
7:     add  $[r]$  as a child of  $p$ 
8:   else
9:     for each child  $k$  of node  $r$  do
10:      MAKE-TWIN-JOINTREE( $\mathcal{T}, k, r$ )
```

(CPT) for endogenous variables. Such a root always exists if we require that each leaf node hosts exactly one family.¹ This defines a single parent for each jointree node $i \neq r$, which is the neighbor of i closest to root r , with all other neighbors of i being its children. The construction procedure is given in Algorithm 4, which is called initially on the base jointree $\mathcal{T}$ with root r and $p = \text{null}$. If all families below r are for root variables, the procedure does not duplicate any nodes. If all families below r are for internal variables, it duplicates the entire subtree rooted at r (including r and its descendants) and adds the duplicate node $[r]$ as a new child of p , where p is the parent of r in the jointree. If none of the above holds, it applies the same procedure recursively to the children of r . Note that when creating a duplicate node $[j]$ for some node j that hosts a family f , we will make $[j]$ host the duplicate family $[f]$, which contains $[X]$ for each variable X in f .

Figure 7.3(b) depicts a (thinned) base jointree constructed for a replication for the base network in Figure 7.3(a). The twin jointree in Figure 7.3(c) is obtained from the base jointree by Algorithm 4, which simply adds nodes and edges to the base jointree. If an edge (i, j) in the base jointree is duplicated by Algorithm 4, we call it a *duplicated edge* and $([i], [j])$ a *duplicate edge*. Otherwise, we call (i, j) an *invariant edge*.

Algorithm 4 constructs the skeleton of the twin jointree hosting the families (CPTs)

¹The standard definition of jointrees allows any node to be a host of any number of families. Our definition facilitates the upcoming treatment and does not preclude optimal jointrees.

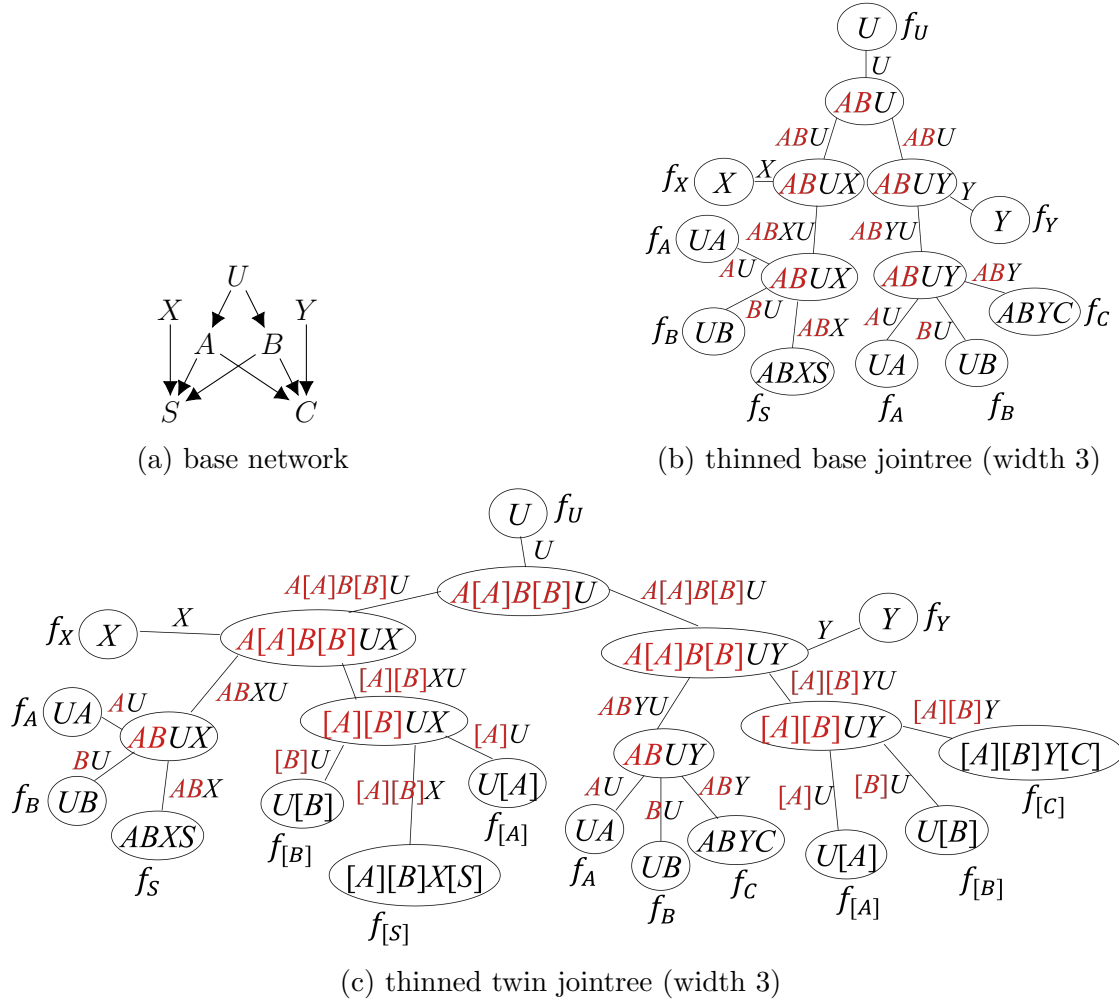


Figure 7.3: Illustrating the construction of a twin jointree from a base jointree.

in the twin network. To obtain a full twin jointree, one still needs to specify the thinned separators and clusters, which is not unique in general; see Chapter 6 for details. We now present a particular set of thinned separators for twin jointrees that can be obtained directly from the separators of base jointrees and that provides complexity guarantees.

Theorem 13. *Consider a twin jointree constructed by Algorithm 4 from a base jointree with thinned separators $\mathbf{S}$ (Definition 14). The following $\mathbf{S}^t$ are valid thinned separators for this twin jointree: for duplicated edges (i, j) , $\mathbf{S}_{ij}^t = \mathbf{S}_{ij}$; for duplicate edges $([i], [j])$; $\mathbf{S}_{[i][j]}^t = [\mathbf{S}_{ij}]$; and for invariant edges (i, j) , $\mathbf{S}_{ij}^t = \mathbf{S}_{ij} \cup [\mathbf{S}_{ij}]$.*

Figure 7.3(c) also depicts the thinned separators and clusters of the twin jointree obtained from those of the base jointree via Theorem 13. This theorem, together with Algorithm 4, provides an efficient way to convert a base jointree into a twin jointree. This is significant for two reasons. First, as pointed out earlier, this method avoids the explicit construction of thinned jointrees for twin networks. Second, we have the following guarantee on the width of twin jointrees constructed by Theorem 13.

Corollary 6. *Consider the base and twin jointrees in Theorem 13. If the base jointree has width w , then the twin jointree has width no greater than $2w + 1$.*

We can now bound the causal treewidth of twin networks.

Corollary 7. *If w and w^t are the causal treewidths of a base network and its twin network, then $w^t \leq 2w + 1$.*

7.3 Counterfactual Reasoning Beyond Two Worlds

A more general setup arises when counterfactual reasoning involves conflicting observations and actions in more than two worlds. In the college admission example, we may be interested in $\Pr(a_s, \bar{a}_{\bar{s}} | \bar{a}, \bar{s})$, which represents the probability that the student would have been admitted had the college assumed everyone had a high standardized score (a_s), and would have been rejected had the college assumed every had a low standardized score ($\bar{a}_{\bar{s}}$), given that the student had a low standardized score and got rejected ($\bar{a}, \bar{s}$). Another scenario is when the states of some exogenous variables change over time. For instance, in system testing, we may provide different input values to the system and identify the causes of abnormal behaviors based on the outputs. When these inputs are modeled as roots in the SCM, we must construct multiple worlds to account for the different values they take across testing cases. For example, Figure 7.4(b) models a scenario where the roots $\{A, B\}$ have different states across three timestamps and therefore contains three copies of them. In general, we

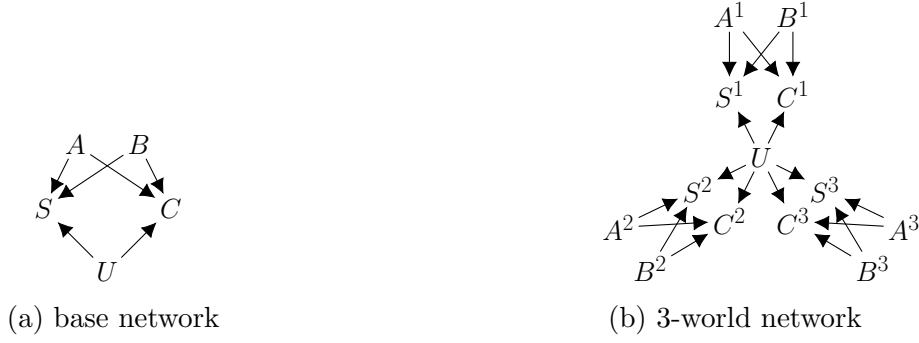


Figure 7.4: A 3-world network where the roots A, B are duplicated.

may create N ($N \geq 1$) duplicates of any variables in the base network, which leads to the notion of *N-world networks* [HCD23].

Definition 18. Consider a base network G and let $\mathbf{R}$ be a subset of its roots and $N \geq 1$ be an integer. The *N-world network* G^N of G is constructed as follows. For each variable X in G that is not in $\mathbf{R}$, replace it with N duplicates of X , labeled $X^1, X^2, \dots, X^N$. For each parent P of X , if P is in $\mathbf{R}$, make P a parent of X^i for all $i \in 1, 2, \dots, N$. Otherwise, make P^i a parent of X^i for all $i \in 1, 2, \dots, N$.

The *parallel worlds model* in [ASP05] is a special case of N -world networks where none of the roots are duplicated; twin networks are a further special case with $N = 2$.

The following result bounds the causal treewidths of N -world networks using the causal treewidths of their base networks.

Theorem 14. If w and w^t are the causal treewidths of a base network and its N -world network, then $w^t \leq N(w + 1) - 1$.

That is, counterfactual reasoning can be accelerated using thinned jointrees developed for marginal inference, and the complexity results in Chapter 6 can be used to bound the complexity of counterfactual reasoning when the number of conflicting worlds (N) is fixed. Note that since classical jointrees are a special case of thinned jointrees when no variables

are functional, all bounds derived in this chapter also apply to classical notions of width and treewidth.

7.4 Beyond Fully Specified SCMs

The results we presented so far on N -world networks, including twin networks, apply to counterfactual reasoning in *fully specified* SCMs. We next discuss how they also apply to *partially specified* SCMs, a more challenging setup in causal reasoning.

A partially specified SCM typically includes the graph structure and some partial information about its parameters. For example, we may not know any of the SCM parameters, or we may know the structural equations but not the distributions over exogenous variables as assumed in [ZAC21]. The identifiability problem becomes central in this setup, and a significant amount of work has focused on characterizing conditions under which the query can be identified; see, e.g., [SP08, CLB21]. When a query is unidentifiable (i.e., no point estimate exists), the classical approach attempts to derive a closed-form bound; see, for example, [BP94a, Pea99, TP00, DMM17, RGW18, Eva18, ZTB22, MLP22].

Some recent work takes a different *learning-based* approach by first estimating the SCM parameters to yield a fully specified SCM, and then using it answer identifiable causal queries through classical inference algorithms [ZAC21, Dar21a, HCA23]. Our results on twin and N -world networks apply directly in this case as they can be used when conducting inference on the fully parameterized SCM. Moreover, when the mechanisms (parameters) for endogenous variables are unknown and need to be learned from data, our methods based on thinned joint-trees can be applied to accelerate the learning process — which involves multiple iterations of inference as in the EM algorithm [DLR77] — since the speedups can be made without fixing the parameters. When the causal query is unidentifiable, the learning-based approach establishes (approximate) bounds by learning multiple sets of parameters and computing the query using each set [ZAC21, HCA23]. Our results are again useful for bounding the

complexity of query computation given each parameter set within this process.

7.5 Experiments

We consider experiments that target random networks whose structures emulate the structures of SCMs used in counterfactual reasoning. We compare the widths of the following types of (thinned) twin jointrees: classical, twin jointrees constructed from the base jointrees using Algorithm 4 (TWIN-DUP); classical, twin jointrees directly constructed from the twin networks (TWIN-MF); thinned, twin jointrees constructed from the base, thinned jointrees using Algorithm 4 and Theorem 13 (TWIN-THINNED-DUP); and thinned, twin jointrees directly constructed from the twin networks (TWIN-THINNED-MF). We also compare them with classical, base jointrees obtained from variable elimination with the minfill heuristic (BASE-MF), and thinned, base jointrees constructed using the procedure in Chapter 6 (BASE-THINNED-MF).²

We generate random networks according to the method used in [Dar20]. Given a number of nodes n and a maximum number of parents p , the method chooses the parents of node X_i randomly from the set $X_1, \dots, X_{i-1}$. The number of parents for node X_i is chosen randomly from the set $0, \dots, \min(p, i - 1)$. We refer to these networks as rNET. We then consider each internal node N and add a unique root as parent for N . This is meant to emulate the structure of SCMs where the additional root can be viewed as representing the different causal mechanisms for endogenous variable N . We refer to these modified networks as rSCM. The twin networks of rSCM are more complex than those for rNET since more variables are shared between the actual and counterfactual worlds.

We consider $n \in \{50, 75, 100, 125, 150, 200, 250, 300\}$ and $p \in \{3, 5, 7\}$. For each combination of n and p , we generate 50 random base networks and report the averages of two properties for the constructed jointrees: *width* and *normalized width*. If a jointree has clus-

²To prevent blowup in replication sizes, we bound the lengths of functional chains by 10.

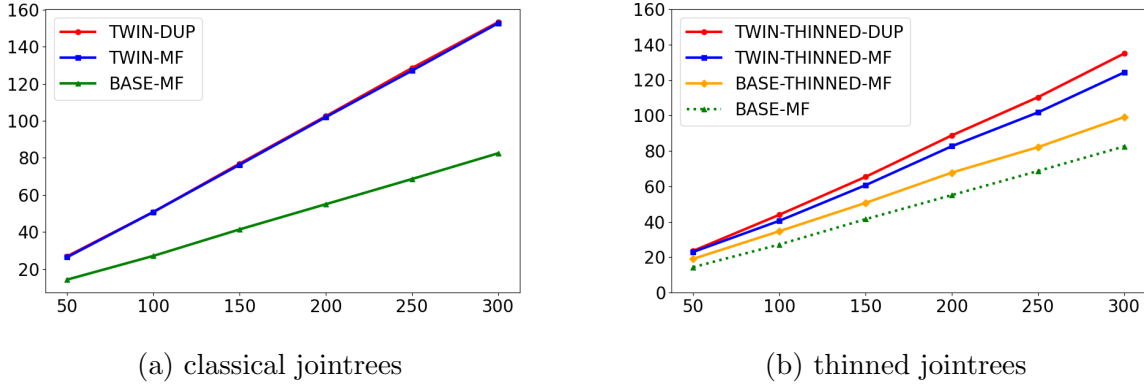


Figure 7.5: Width of jointtrees (y-axis) against number of variables in the base network (x-axis) for maximum number of parents $p = 5$.

ters $\mathbf{C}_1, \dots, \mathbf{C}_n$, then its normalized width is $\log_2 \sum_{i=1}^n 2^{|\mathbf{C}_i|}$. This accounts for all clusters in the jointtree (instead of just the largest one) and the jointtree size. Detailed experimental results can be found in [HCD23], which also includes experiments using random networks generated according to the method in [IC02]. We provide representative plots in Figure 7.5 for jointtree widths under $p = 5$ and discuss patterns exhibited in the results.³

First, the widths of twin jointtrees are always less than twice the widths of their base jointtrees and often significantly less than that. This is not guaranteed by our theoretical bounds as the jointtree widths depend on the quality of heuristics. Second, twin jointtrees constructed by directly applying Algorithm 4 to a base jointtree (TWIN-DUP) have similar widths as those constructed by operating on the twin network (TWIN-MF). This also holds for thinned jointtrees (TWIN-THINNED-DUP vs TWIN-THINNED-MF) and is encouraging since the former methods are much more efficient than the latter ones. Third, the employment of thinned jointtrees can lead to significant reduction in width and hence an exponential reduction in computations. This can be seen by comparing the widths of twin jointtrees TWIN-THINNED-DUP and TWIN-DUP since the former is thinned but the latter

³One interesting observation from the results is that the BASE-MF is better than BASE-THINNED-MF for rSCMs. This suggests imperfections in our replication strategy and the thinning heuristics for constructing thinned jointtrees in [CD22], since the optimal width of thinned jointtrees should not exceed that of jointtrees.

is not (similarly for TWIN-THINNED-MF and TWIN-MF). Finally, consider Figure 7.5(b) for a bottom-line comparison between the complexity of counterfactual reasoning and the complexity of associational/interventional reasoning in practice. Jointrees BASE-MF have the smallest widths for base networks so these would be used for associational/interventional reasoning. The best twin jointrees are TWIN-THINNED-MF so these are what one would use for counterfactual reasoning. The widths of latter jointrees are always less than twice the widths of the former, and quite often significantly much less.

CHAPTER 8

Conclusion

We investigated the role of background knowledge in causal inference and showed that incorporating knowledge (parametric constraints) provides a promising avenue for making causal inference more practical. In summary, we demonstrated two advantages enabled by incorporating background knowledge, both of which address core challenges in the field.

The first advantage is on improving the identifiability of causal queries. This improvement is especially important as without establishing identifiability, causal inference may not even be feasible. We focused on the impact of knowledge on the identifiability of causal effects, a common class of causal queries, and showed that knowledge serves as additional information that can be leveraged to render more causal effects identifiable. We started by formulating knowledge as additional restrictions on the set of causal models considered in the identifiability problem. We examined the impact of multiple types of parametric constraints, including functional dependencies, context-specific independencies, and state constraints, and developed methodologies dedicated to incorporating each. These methods generally exploit constraints to simplify causal graphs, enabling the identification of more causal effects. In addition, we developed a systematic treatment using tractable arithmetic circuits that integrates various types of constraints in a coherent manner.

Despite the efforts made in this dissertation and elsewhere, this direction remains in its early stages, as vast forms of knowledge (e.g., logical constraints) and causal queries (e.g., conditional causal effects) have yet to be explored. A further complication is the interplay among different sources of information, including both knowledge and data, as highlighted by

the interaction between functional dependencies and positivity constraints in Chapter 3. We believe progress in this direction will require moving beyond studying each form of knowledge separately toward developing a unified framework that integrates different forms of data and knowledge to maximize identifiability, with the arithmetic circuit approach in Chapter 5 serving as an initial step.

The second advantage concerns the computational efficiency of causal queries once they are deemed identifiable. A natural starting point — and the main setup of this dissertation — is when causal models are fully specified, under which all causal queries are identifiable and can be reduced to marginal queries. We showed that background knowledge can be exploited to accelerate marginal computations beyond the complexity barrier imposed by classical inference methods. In particular, we leveraged functional dependencies to develop thinned jointrees and the associated complexity notion of causal treewidth, enabling significant computational speedups. These gains carry over directly to associational and interventional reasoning, and further improve counterfactual reasoning by bounding its complexity via causal treewidth, as demonstrated both theoretically and empirically in Chapter 7.

Beyond this initial effort on fully specified models, it is crucial to extend the results to more general scenarios involving partially specified models. Classical approaches have largely relied on deriving closed-form formulas to estimate causal queries. These formulas, however, can be cumbersome and expensive to compute, if feasible at all; see [DRT25, TK17b] for some recent efforts. Finding formulas that can be computed efficiently remains an ongoing research topic, and we hope that incorporating knowledge will help address this in the future. On the other hand, the learning-based approach shown in Chapter 7 avoids the evaluation of formulas, yet its estimates are generally approximate, with accuracy depending on the quality of parameter learning and the number of runs. In this dissertation, we have discussed the benefits of incorporating knowledge to improve the computational efficiency of this approach; however, its impact on estimation accuracy remains to be investigated.

APPENDIX A

Appendix for Chapter 2

A.1 Projection

The projection [Ver93, TP02b, TP03] of G onto $\mathbf{V}$ constructs a new DAG G' over variables $\mathbf{V}$ as follows. Initially, DAG G' contains variables $\mathbf{V}$ but no edges. Then, for every pair of variables $X, Y \in \mathbf{V}$, an edge is added from X to Y to G' if X is a parent of Y in G or if there exists a directed path from X to Y in G such that none of the internal nodes on the path is in $\mathbf{V}$. Furthermore, a bidirected edge $X \longleftrightarrow Y$ is added between every pair of variables X and Y in G' if there exists a divergent path.¹ For example, the projection of the DAG in Figure 2.1(a) onto A, C, E, X_1, X_2, Y yields Figure 2.1(c). A bidirected edge $X \longleftrightarrow Y$ is a compact notation for $X \leftarrow H \rightarrow Y$, where H is an auxiliary hidden variable. Hence, the projected DAG in Figure 2.1c can be interpreted as a classical DAG but with additional, hidden root variables.

The projection operation is guaranteed to produce a DAG G' in which hidden variables are all roots and each has exactly two children. Graphs that satisfy this property are called *semi-Markovian* and can be fed as inputs to the ID or IDENTIFY algorithm for testing of identifiability [TP03, SP06]. Moreover, projection preserves some key properties of G , such as d-separation [Ver93] among $\mathbf{V}$, which guarantees that identifiability is preserved when working with G' instead of G [TP02b].

¹A divergent path between X and Y is a path in the form of $X \leftarrow \dots \leftarrow U \rightarrow \dots \rightarrow Y$ between X and Y in G such that none of the internal nodes on the path is in $\mathbf{V}$.

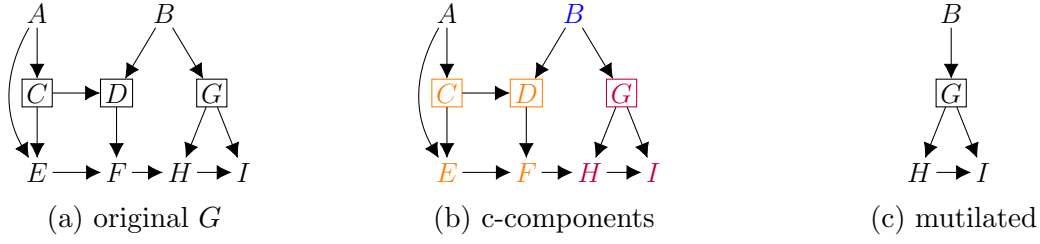


Figure A.1: The causal graph G , its mutilated graph under $\Pr_F(I)$, and the c-component decomposition. Hidden variables are shown in rectangles, and variables in different c-components are highlighted with different colors.

A.2 Background on C-Components

Sound and complete methods for testing classical identifiability based on causal graphs have been developed in the past; see, e.g., IDENTIFY [TP03, HV06] and ID [SP06]. These methods all rely on a key graphical notion called *c-component*, which we review next.

Definition 19 (c-component [TP02b]). *Let G be a causal graph with a subset of observed variables. A variable set $\mathbf{C}$ is said to be a c-component in G if it contains a single observed variable with no hidden parents or satisfies the following conditions: (i) every pair of hidden variables in $\mathbf{C}$ are either adjacent or have a common child; and (ii) every observed variable in $\mathbf{C}$ has some hidden parent in $\mathbf{C}$.²*

A c-component is *maximal* if it is not a subset of any other c-component. We can always *partition* the variables (both observed and hidden) in a causal graph into maximal c-components. For example, Figure A.1(b) depicts the c-component partition of the causal graph, where variables in different c-components are highlighted with different colors.

Moreover, each c-component $\mathbf{C}$ induces a *c-factor* defined as $Q[\mathbf{C}] = \sum_{\mathbf{H}} \prod_{C \in \mathbf{C}} \theta_{C|\mathbf{P}_C}$, where $\mathbf{H}$ denotes the hidden variables in $\mathbf{C}$ and $\mathbf{P}_C$ denotes the parents of C in G . Based on the definition, the c-factor $Q[\mathbf{C}]$ is over all observed variables appearing in the CPTs of $\mathbf{C}$.

²[TP02b] only includes observed variables in c-components; we also include the hidden variables so that c-factors can be defined directly from the c-components. Note also that the definition of c-components here is for causal Bayesian networks, which is more general than the one for semi-Markovian (projected) graphs.

To illustrate, the c-factor for the c-component $\{C, D, E, F\}$ in Figure A.1(b) is computed as $Q[CDEF] = \sum_{CD} \theta_{C|A} \theta_{D|BC} \theta_{E|AC} \theta_{F|DE}$, which is over variables $\{A, B, E, F\}$. One important property of c-factors is that they decompose the observational distribution $\Pr(\mathbf{V})$ as follows. Let $\{\mathbf{C}_1, \dots, \mathbf{C}_m\}$ be the c-components that partition variables in G , then $\Pr(\mathbf{V}) = \prod_{i=1}^m Q[\mathbf{C}_i]$ [TP03]. In Figure A.1(b), for example, $\Pr(\mathbf{V}) = Q[A]Q[B]Q[CDEF]Q[GHI]$.

A.3 Causal Effect Identification Algorithms

After obtaining a projected causal graph, we can apply both the ID [SP06] and IDENTIFY [TP02b] algorithms for identifying the causal effects under the strict positivity assumption $\Pr(\mathbf{V}) > 0$. Both algorithms utilize the notion of c-components (Appendix A.2) and return an identifying formula if the causal effect is identifiable, or FAIL otherwise. The algorithms are both sound and complete, and are therefore equivalent in their ability to test identifiability. Specifically, if the algorithms return FAIL, the causal graph must contain a *hedge*, a graphical structure that induces the unidentifiability [SP06].

We focus here on the IDENTIFY algorithm [TP03], as its structure is used to develop algorithms for testing causal effects under context-specific independencies in Chapter 4. Intuitively, the IDENTIFY algorithm converts the problem of identifying causal effects to independent problems of identifying c-factors. In particular, a c-factor is identifiable iff its entries are uniquely determined (computable) by the observational distribution $\Pr(\mathbf{V})$. Consider a causal graph G , treatments $\mathbf{X}$, and outcomes $\mathbf{Y}$. Let T be the mutilated graph obtained by first removing treatment variables $\mathbf{X}$ from G and then retaining only the ancestors of $\mathbf{Y}$ (Figure A.1(c) depicts the mutilated graph of Figure A.1(a)). The algorithm then reduces the identifiability of the causal effect $\Pr_{\mathbf{X}}(\mathbf{Y})$ to that of c-factors induced by the c-components in T . This reduction is both sound and complete [HV06, SP06]: the causal effect is identifiable iff all the c-factors are identifiable. The algorithm then checks the identifiability of c-factors using graph operations on the c-components; see [TP03] for more details.

APPENDIX B

Appendix for Chapter 3

B.1 Functional Elimination for Causal Bayesian Networks

The functional elimination in Definition 6 removes functional variables from a DAG G and yields another DAG G' on the remaining variables. We have shown in the main dissertation that the functional elimination preserves the D-separations. Here we extend the notion of functional elimination to Causal Bayesian networks (CBNs) which contain not only a causal graph (DAG) but also CPTs. We show that the (extended) functional elimination preserves the marginal distribution on the remaining variables. That is, given any CBN with the causal graph G , we can construct another CBN with the causal graph G' such that the two CBNs induce a same distribution on the variables in G' , where G' is the result of eliminating functional variables from G . Moreover, we show that the functional elimination operation further preserves the causal effects, which makes it applicable to the causal identification. This extended version of functional elimination and the corresponding results will be used for the proofs.

Recall that a CBN $\langle G, \mathcal{F} \rangle$ contains a causal graph G and a set of CPTs (factors) $\mathcal{F}$. We first extend the definition of functional elimination (Definition 6) from DAGs to CBNs. We will use f_X to denote the CPT for variable X in $\mathcal{F}$.

Definition 20. *The functional elimination of a functional variable X from a CBN $\langle G, \mathcal{F} \rangle$ yields another CBN $\langle G', \mathcal{F}' \rangle$ obtained as follows. The DAG G' is obtained from G by Definition 6. For each child C of X , its CPT in $\mathcal{F}'$ is $(\sum_X f_X f_C)$ where f_X, f_C are the*

corresponding CPTs in $\mathcal{F}$.

We first show that the new CPTs produced by Definition 20 are well-defined.

Proposition 9. *Let f_X and f_Y be the CPTs for variables X and Y in a CBN, then $(\sum_X f_X f_Y)$ is a valid CPT for Y .*

The next proposition shows that functional elimination preserves the functional dependencies.

Proposition 10. *Let $\mathcal{M}'$ be the CBN resulting from functionally eliminating a functional variable from a CBN $\mathcal{M}$. Then each variable (from $\mathcal{M}'$) is functional in $\mathcal{M}'$ if it is functional in $\mathcal{M}$.*

The next theorem shows that the order of functional elimination does not matter.

Proposition 11. *Let $\mathcal{M}$ be a CBN and π_1, π_2 be two variable orders over a set of functional variables $\mathbf{W}$. Then functionally eliminating $\mathbf{W}$ from $\mathcal{M}$ according to π_1 and π_2 yield the same CBN.*

The next result shows that eliminating functional variables preserves the marginal distribution.

Theorem 15. *Consider a CBN $\mathcal{M}$ which induces Pr . Let $\mathcal{M}'$ be the result of functionally eliminating a set of functional variables $\mathbf{W}$ from $\mathcal{M}$ which induces Pr' . Then $\text{Pr}' = \sum_{\mathbf{W}} \text{Pr}$.*

One key property of functional elimination is that it preserves the interventional distribution over the remaining variables. This property allows us to eliminate functional variables from a causal graph and estimate the causal effects in the resulting graph.

Theorem 16. *Let $\mathcal{M}'$ be the CBN over variables $\mathbf{V}$ resulting from functionally eliminating a set of functional variables $\mathbf{W}$ from a CBN $\mathcal{M}$. Then $\mathcal{M}'$ and $\mathcal{M}$ attain the same $\text{Pr}_{\mathbf{x}}(\mathbf{V})$ for any $\mathbf{X} \subseteq \mathbf{V}$.*

B.2 Proofs

Proof of Proposition 1. Our goal is to construct two different parameterizations $\mathcal{F}^1$ and $\mathcal{F}^2$ that induce the same $\Pr(\mathbf{V})$ but different $\Pr_{\mathbf{x}}(\mathbf{y})$. This is done by first creating a parameterization $\mathcal{F}$ which contains strictly positive CPTs for all variables, and then constructing $\mathcal{F}^1$ and $\mathcal{F}^2$ based on $\mathcal{F}$.

Let $\mathcal{P}$ be the directed path from $X = X_i$ to Y , denoted $X \rightarrow Z \rightarrow \dots \rightarrow Y$ which does not contain any treatment variables other than X . Let $\mathbf{P}_X$ be the parents of X in G . For each node M on the path, let $\mathbf{P}_M$ be the parents of M except for the parent that lies on $\mathcal{P}$. Moreover, for each variable M on $\mathcal{P}$, we will only modify the conditional probability for a single state m^* of M , where $x^* \in \mathbf{x}$ is the treated state of X . Let ϵ be an arbitrarily small constant (close to 0), we next show the modifications for the CPTs in $\mathcal{F}^1$.

$$f^1(x|\mathbf{p}_X) = \begin{cases} 0 & \text{if } x = x^* \\ 1/(|X| - 1), & \text{otherwise} \end{cases}$$

$$f^1(z|x, \mathbf{p}_Z) = \begin{cases} 1 - \epsilon, & \text{if } x = x^*, z = z^* \\ \epsilon/(|Z| - 1), & \text{if } x = x^*, z \neq z^* \\ \epsilon, & \text{if } x \neq x^*, z = z^* \\ (1 - \epsilon)/(|Z| - 1), & \text{if } x \neq x^*, z \neq z^* \end{cases}$$

For every variable $T \notin \{X, Z\}$ which has parent Q on the path $\mathcal{P}$, we assign

$$f^1(t|q, \mathbf{p}_T) = \begin{cases} 1 - \epsilon, & \text{if } q = q^*, t = t^* \\ \epsilon/(|T| - 1), & \text{if } q = q^*, t \neq t^* \\ \epsilon, & \text{if } q \neq q^*, t = t^* \\ (1 - \epsilon)/(|T| - 1), & \text{if } q \neq q^*, t \neq t^* \end{cases}$$

We assign the same CPTs for X and all variables $T \notin \{X, Z\}$ but a different CPT for Z in $\mathcal{F}^2$.

$$f^2(z|x, \mathbf{p}_Z) = \begin{cases} \epsilon, & \text{if } x = x^*, z = z^* \\ (1 - \epsilon)/(|Z| - 1), & \text{if } x = x^*, z \neq z^* \\ \epsilon, & \text{if } x \neq x^*, z = z^* \\ (1 - \epsilon)/(|Z| - 1), & \text{if } x \neq x^*, z \neq z^* \end{cases}$$

The two parameterizations $\mathcal{F}^1$ and $\mathcal{F}^2$ induce the same $\Pr(\mathbf{V})$ where $\Pr(\mathbf{v}) = 0$ if $x^* \in \mathbf{v}$ and $\Pr(\mathbf{v}) > 0$ otherwise. We next show that the parameterization satisfies each positivity constraint $\Pr(\mathbf{S}|\mathbf{Z})$ as long as it does not imply $\Pr(X) > 0$. We first show that $X \in \mathbf{S}$ implies $\Pr(X) > 0$. This is because $\Pr(\mathbf{S}) = \sum_{\mathbf{z}} \Pr(\mathbf{S}|\mathbf{z}) \Pr(\mathbf{z})$ and there must exist some instantiation $\mathbf{z}$ where $\Pr(\mathbf{z}) > 0$ and $\Pr(\mathbf{S}|\mathbf{z}) > 0$ by constraint. This implies $\Pr(\mathbf{S}) > 0$ and therefore $\Pr(X) > 0$. Hence, $\mathcal{C}_{\mathbf{V}}$ does not contain such constraint $\Pr(\mathbf{S}|\mathbf{Z})$ where $X \in \mathbf{S}$. Suppose $X \in \mathbf{Z}$, then $\Pr(\mathbf{z}) > 0$ if and only if $x^* \notin \mathbf{z}$. Moreover, since $\Pr(\mathbf{v}) > 0$ whenever $x^* \notin \mathbf{v}$, it is guaranteed that $\Pr(\mathbf{S}, \mathbf{z}) > 0$ when $\Pr(\mathbf{z}) > 0$, which implies $\Pr(\mathbf{S}|\mathbf{Z}) > 0$. Finally, suppose $X \notin (\mathbf{S} \cup \mathbf{Z})$, then $\Pr(\mathbf{S}, \mathbf{Z}) = \sum_x \Pr(\mathbf{S}, \mathbf{Z}, x) > 0$. Hence, the positivity constraint is satisfied by both parameterizations. By construction, $\Pr^1$ and $\Pr^2$ induce different values for the causal effect $\Pr_{\mathbf{x}}(\mathbf{y})$ since the probability of $Y = y^*$ under the treatment $do(X = x^*)$ will be different for the two parameterizations. $\square$

Proposition 12. *Let G be a causal graph and $\mathbf{V}$ be its observed variables. A set of functional variables $\mathbf{W}$ is consistent with positivity constraints $\mathcal{C}_{\mathbf{V}}$ if no constraint in $\mathcal{C}_{\mathbf{V}}$ mentions both $W \in \mathbf{W}$ and a set $\mathbf{H}_W$ that intercepts all directed paths from non-functional variables to W .*

Proof of Proposition 12. We construct a parameterization (CPTs) $\mathcal{F}$ and show that the distribution $\Pr$ induced by $\mathcal{F}$ satisfies the $\mathcal{C}_{\mathbf{V}}$, which ensures the consistency. The states of each variable V are represented in the form of $(s_V, p_1, \dots, p_m)$ where s_V and p_i ($i \in \{1, \dots, m\}$) are all binary indicator (0 or 1). Specifically, each of the p_i corresponds to a “functional descendant paths” of V defined as follows: a functional descendant path of V is a directed

path that starts with V and that all variables on the path (excluding V) are functional. Suppose V does not have any functional descendant paths, then the states of V is simply represented as (s_V) .

We next show how to assign CPTs for each variable in the causal graph G based on whether the variable is functional. For each non-functional variable, we assign a uniform distribution. For each functional variable W whose parents are $T_1, \dots, T_n$ and whose functional descendant paths are $\mathcal{P}_1, \dots, \mathcal{P}_m$, we assign the CPT f_W as follows:

$$\begin{aligned}
s_W &\leftarrow p_{Ind(T_1, W)}^{T_1} \oplus \dots \oplus p_{Ind(T_n, W)}^{T_n} \\
p_1 &\leftarrow p_{1,1}^{T_1} \oplus \dots \oplus p_{n,1}^{T_n} \\
&\dots \\
p_m &\leftarrow p_{1,m}^{T_1} \oplus \dots \oplus p_{n,m}^{T_n}
\end{aligned} \tag{B.1}$$

where $Ind(T_i, W)$ denotes the index assigned to the path $\{(T_i, W)\}$ (which contains a single edge) in the state of T^i , and $p_{i,j}^{T_i}$ denotes the indicator in the state of T_i for the functional descendant path $\mathcal{P}'$ that contains the functional descendant path $\mathcal{P}_j$, i.e., $\mathcal{P}' = \{(T_i, W)\} \cup \mathcal{P}_j$.

For simplicity, we call the set of variables $\mathbf{H}_W$ that satisfies the condition in the proposition a “functional ancestor set” of W . We show that $\Pr(\mathbf{S}, \mathbf{Z}) > 0$ for each positivity constraint in the form of $\Pr(\mathbf{S}|\mathbf{Z}) > 0$. Let $\mathbf{W} \subseteq \mathbf{S} \cup \mathbf{Z}$ be a subset of functional variables. Since $\mathbf{S} \cup \mathbf{Z}$ does not contain any functional ancestor set of W for each $W \in \mathbf{W}$, it follows that there exist directed paths from a set of non-functional variables $\mathcal{A}'_W$ to W that are unblocked by $\mathbf{M} = \mathbf{S} \cup \mathbf{Z} \setminus \{W\}$ and contain only functional variables (excluding $\mathcal{A}'_W$). We can further assume that $\mathcal{A}'_W$ is chosen such that the set $\mathcal{A}_W = \mathbf{M} \cup \mathcal{A}'_W$ forms a valid functional ancestor set for W . We next show that for any state w of W and instantiation $\mathbf{m}$ of $\mathbf{M}$, there exists at least one instantiation $\mathbf{a}$ of $\mathcal{A}'_W$ such that $\Pr(w, \mathbf{m}, \mathbf{a}) > 0$.

Let $\mathbf{P}^W$ denote the set of all directed paths from $\mathcal{A}_W$ to W that do not contain $\mathcal{A}_W$ (except for the first node on the path). Let $\mathbf{P}_1^W \subseteq \mathbf{P}^W$ be the paths that start with a

variable in $\mathbf{M}$, and $\mathbf{P}_2^W \subseteq \mathbf{P}^W$ be other paths that start with a variable in $\mathcal{A}'_W$. Moreover, for any path $\mathcal{P}$, let $\text{pathval}(\mathcal{P})$ be the binary indicator (e.g., p_1) for $\mathcal{P}$ in the state of $\mathcal{P}(0)$ (first variable in $\mathcal{P}$). Since the value assignments for $\text{pathval}(\mathcal{P})$ are independent for different $\mathcal{P}$'s, we can always find some instantiation $\mathbf{a} \in \mathcal{A}'_W$ such that the following equality holds given w and $\mathbf{m}$:

$$\bigoplus_{\mathcal{P}_2 \in \mathbf{P}_2^W} \text{pathval}(\mathcal{P}_2) = s_W \oplus \bigoplus_{\mathcal{P}_1 \in \mathbf{P}_1^W} \text{pathval}(\mathcal{P}_1)$$

We next assign values for other path indicators of $\mathbf{a}$ such that the indicators for the functional descendant paths in the state w are set correctly. In particular, for each functional descendant path $\mathcal{P}$ of W , let $\mathbf{P}$ be the set of functional descendant paths of $\mathcal{A}_W$ that do not contain $\mathcal{A}_W$ (except for the first node on the path) and that contain $\mathcal{P}$ as a sub-path. Let $\mathbf{P}_1 \subseteq \mathbf{P}$ be the paths that start with a variable in $\mathbf{M}$, and $\mathbf{P}_2 \subseteq \mathbf{P}$ be other paths that start with a variable in $\mathcal{A}'_W$. Again, since all the indicators for paths in $\mathcal{P}$ are independent, we can assign the indicators for $\mathbf{a} \in \mathcal{A}'_W$ such that

$$\bigoplus_{\mathcal{P}_2 \in \mathbf{P}_2} \text{pathval}(\mathcal{P}_2) = \text{pathval}(\mathcal{P}) \oplus \bigoplus_{\mathcal{P}_1 \in \mathbf{P}_1} \text{pathval}(\mathcal{P}_1)$$

Finally, we combine the cases for each individual $W \in \mathbf{W}$ by creating the following set $\mathcal{A}_{\mathbf{W}} = \bigcup_{W \in \mathbf{W}} \mathcal{A}_W$. Since all the functional descendant paths we considered for different W 's are disjoint, we can always find an assignment $\mathbf{a}$ for $\mathcal{A}_{\mathbf{W}}$ that is consistent with the functional dependencies (does not produce any zero probabilities). Consequently, there must exist some full instantiation $(\mathbf{u}, \mathbf{v})$ compatible with $\mathbf{s}, \mathbf{z}, \mathbf{a}$ such that $\Pr(\mathbf{u}, \mathbf{v}) > 0$, which implies $\Pr(\mathbf{s}, \mathbf{z}) > 0$. $\square$

Proof of Proposition 9. Suppose Y is not a child of X in the CBN, then $\sum_X f_X f_Y = f_Y(\sum_X f_X)$ which is guaranteed to be a CPT for Y . Suppose Y is a child of X . Let $\mathbf{P}_X$ denote the parents of X and $\mathbf{P}_Y$ denote the parents of Y excluding X . The new factor

$g = \sum_X f_X f_Y$ is defined over $\mathbf{P}_X \cup \mathbf{P}_Y \cup \{Y\}$. Consider each instantiation $\mathbf{p}_X$ and $\mathbf{p}_Y$, then $\sum_y g(\mathbf{p}_X, \mathbf{p}_Y, y) = \sum_y \sum_x f_X(x|\mathbf{p}_X) f_Y(y|\mathbf{p}_Y, x) = \sum_x f_X(x|\mathbf{p}_X) \sum_y f_Y(y|\mathbf{p}_Y, x) = 1$. Hence, g is a CPT for Y . $\square$

Proof of Proposition 10. Let X be the functional variables that is functionally eliminated. By definition, the elimination only affects the CPTs for the children of X . Hence, any functional variable that is not a child of X remains functional. For each child C of X that is functional, the new CPT $(\sum_X f_X f_C)$ only contains values that are either 0 or 1 since both f_X and f_C are functional. $\square$

Proof of Proposition 11. First note that π_2 can always be obtained from π_1 by a sequence of “transpositions”, where each transposition swaps two adjacent variables in the first sequence. Let π be an elimination order and let π' be the elimination order resulted from swapping $\pi_i = X$ and $\pi_{i+1} = Y$ from the π , i.e.,

$$\pi = (\dots, X, Y, \dots) \quad \pi' = (\dots, Y, X, \dots)$$

We show functional elimination according to π and π' yield a same CBN, which can be applied inductively to conclude that elimination according to π_1 and π_2 yield a same CBN. Since π and π' agree on a same elimination order up to X , they yield a same CBN before eliminating variables X, Y . It suffices to show the CBNs resulting from eliminating (X, Y) and eliminating (Y, X) are the same. Let $\langle G, \text{Pr} \rangle$ be the CBN before eliminating variables X, Y . Suppose X and Y do not belong to a same family (which contains a variable and its parents), the elimination of X and Y are independent and the order of elimination does not matter. Suppose X and Y belong to a same family, then they are either parent and child or co-parents.¹

WLG, suppose X is a parent of Y . Eliminating (X, Y) and eliminating (Y, X) yield a same causal graph that is defined as follows. Each child C of Y has parents $\mathbf{P}_X \cup \mathbf{P}_Y \cup$

¹ X and Y are co-parents if they have a same child.

$\mathbf{P}_C \setminus \{X, Y\}$, and any other child C of X has parents $\mathbf{P}_X \cup \mathbf{P}_C \setminus \{X\}$. We next consider the CPTs. For each common child C of X and Y , its CPT resulting from eliminating (X, Y) is $f_C^1 = \sum_Y (\sum_X f_C f_X) (\sum_X f_Y f_X)$, and the CPT resulting from eliminating (Y, X) is $f_C^2 = \sum_X f_X (\sum_Y f_C f_Y)$. Since X is a parent of Y , we have $Y \notin f_X$ and

$$f_C^2 = \sum_X \sum_Y f_X f_C f_Y = \sum_Y \sum_X f_X f_C f_Y = \sum_Y (\sum_X f_X f_C) (\sum_X f_X f_Y) = f_C^1$$

We next consider the case when C is a child of Y but not a child of X . The CPT for C resulting from eliminating (X, Y) is $f_C^1 = \sum_Y f_C (\sum_X f_Y f_X)$, and the CPT resulting from eliminating (Y, X) is $f_C^2 = \sum_X f_X (\sum_Y f_C f_Y)$. Again, since X is a parent of Y , we have $Y \notin f_X$ and

$$f_C^2 = \sum_X \sum_Y f_X f_C f_Y = \sum_Y \sum_X f_X f_C f_Y = \sum_Y f_C (\sum_X f_X f_Y) = f_C^1$$

We finally consider the case when C is a child of X but not a child of Y . Regardless of the order on X and Y , the CPT for C resulting from eliminating X and Y is $(\sum_X f_X f_C)$.

We next consider the case when X and Y are co-parents. Regardless of the order on X and Y , the causal graph resulting from the elimination satisfies the following properties: (1) for each common child C of X and Y , the parents for C are $\mathbf{P}_X \cup \mathbf{P}_Y \cup \mathbf{P}_C \setminus \{X, Y\}$; (2) for each C that is a child of X but not a child of Y , the parents for C are $\mathbf{P}_X \cup \mathbf{P}_C \setminus \{X\}$; (3) for each C that is a child of Y but not a child of X , the parents for C are $\mathbf{P}_Y \cup \mathbf{P}_C \setminus \{Y\}$. We next consider the CPTs. For each common child C of X and Y , its CPT resulting from eliminating (X, Y) is $f_C^1 = \sum_Y f_Y (\sum_X f_X f_C)$, and the CPT resulting from eliminating (Y, X) is $f_C^2 = \sum_X f_X (\sum_Y f_Y f_C)$. Since X and Y are not parent and child, we have $X \notin f_Y$, $Y \notin f_X$ and

$$f_C^1 = \sum_Y \sum_X f_Y f_X f_C = \sum_X \sum_Y f_Y f_X f_C = f_C^2$$

For each C that is a child of X but not a child of Y , regardless of the order on X and Y ,

the CPT for C resulting from eliminating variables X and Y is $(\sum_X f_X f_C)$. A similar result holds for each C that is a child of Y but not a child of X . $\square$

Proof of Theorem 15. It suffices to show that $\text{Pr}' = \sum_X \text{Pr}$ when we eliminate a single variable X . Let $\mathcal{F}$ denote the set of CPTs for $\mathcal{M}$. Since f_X is a functional CPT for X , we can replicate f_X in $\mathcal{F}$ which yields a new CPT set (replication) $\mathcal{F}'$ that induce a same distribution as $\mathcal{F}$; see details in [Dar20, Theorem 4]. Specifically, we pair the CPT for each child C of X with an extra copy of f_X , denoted (f_X, f_C) , which yields a list of pairs $(f_X, f_{C_1}), \dots, (f_X, f_{C_k})$ where $C_1, \dots, C_k$ are the children of X . Functionally eliminating X from $\mathcal{F}'$ yields

$$\begin{aligned} \sum_X \text{Pr} &= \sum_X \mathcal{F}' = \mathcal{H} \cdot \left(\sum_X f_X f_{C_1} \right) \cdots \left(\sum_X f_X f_{C_k} \right) \quad [\text{Dar20, Corollary 1}] \\ &= \mathcal{H} \cdot f'_{C_1} \cdots f'_{C_k} = \text{Pr}' \end{aligned} \tag{B.2}$$

where $\mathcal{H}$ are the CPTs in $\mathcal{F}$ that do not contain X and each f'_{C_i} is the CPT for child C_i in $\mathcal{M}'$. $\square$

The following lemma is needed to prove Theorem 16.

Lemma 1. *Consider a CBN $\langle G, \mathcal{F} \rangle$ and its mutilated CBN $\langle G_{\mathbf{x}}, \mathcal{F}_{\mathbf{x}} \rangle$ under $\text{do}(\mathbf{x})$. Let W be a functional variable not in $\mathbf{X}$ and let $\langle G', \mathcal{F}' \rangle$ and $\langle G'_{\mathbf{x}}, \mathcal{F}'_{\mathbf{x}} \rangle$ be the results of functionally eliminating W from $\langle G, \mathcal{F} \rangle$ and $\langle G_{\mathbf{x}}, \mathcal{F}_{\mathbf{x}} \rangle$, respectively. Then $\langle G'_{\mathbf{x}}, \mathcal{F}'_{\mathbf{x}} \rangle$ is the mutilated CBN for $\langle G', \mathcal{F}' \rangle$.*

Proof. First observe that the children of W in G and $G_{\mathbf{x}}$ can only differ by the variables in $\mathbf{X}$. Let $\mathbf{C}_1$ be the children of W in both G and $G_{\mathbf{x}}$ and let $\mathbf{C}_2$ be the children of W in G but not in $G_{\mathbf{x}}$. By the definition of mutilated CBN, W has the same set of parents and CPT in $\langle G, \mathcal{F} \rangle$ and $\langle G_{\mathbf{x}}, \mathcal{F}_{\mathbf{x}} \rangle$. Similarly, each child $C \in \mathbf{C}_1$ has the same set of parents and CPT in $\langle G, \mathcal{F} \rangle$ and $\langle G_{\mathbf{x}}, \mathcal{F}_{\mathbf{x}} \rangle$. Hence, eliminating W yields the same set of parents and CPT for each $C \in \mathbf{C}$ in $\langle G', \mathcal{F}' \rangle$ and $\langle G'_{\mathbf{x}}, \mathcal{F}'_{\mathbf{x}} \rangle$. We next consider the set of parents and CPT for each child

$C \in \mathbf{C}_2$. Since W is not a parent of C in $\langle G_{\mathbf{x}}, \mathcal{F}_{\mathbf{x}} \rangle$, variable C has the same set of parents and CPT in $\langle G'_{\mathbf{x}}, \mathcal{F}'_{\mathbf{x}} \rangle$. The exactly same set of parents (empty) and CPT will be assigned to C in the mutilated CBN for $\langle G', \mathcal{F}' \rangle$. $\square$

Proof of Theorem 16. Consider a CBN $\langle G, \mathcal{F} \rangle$ and its mutilated CBN $\langle G_{\mathbf{x}}, \mathcal{F}_{\mathbf{x}} \rangle$. Let Pr and $\text{Pr}_{\mathbf{x}}$ be the distributions induced by $\mathcal{F}$ and $\mathcal{F}_{\mathbf{x}}$ over variables $\mathbf{V}$, respectively. By Lemma 1, we can eliminate each $W \in \mathbf{W}$ inductively from $\langle G, \mathcal{F} \rangle$ and $\langle G_{\mathbf{x}}, \mathcal{F}_{\mathbf{x}} \rangle$ and obtain $\langle G', \mathcal{F}' \rangle$ and its mutilated CBN $\langle G'_{\mathbf{x}}, \mathcal{F}'_{\mathbf{x}} \rangle$. By Theorem 15, the distribution induced by $\mathcal{F}'_{\mathbf{x}}$ is exactly $\sum_{\mathbf{w}} \text{Pr}_{\mathbf{x}}(\mathbf{V})$. $\square$

Proof of Proposition 2. First note that the extended set $\mathbf{Z}'$ contains $\mathbf{Z}$ and all variable that are functionally determined by $\mathbf{Z}$. Consider any path $\mathcal{P}$ between some $X \in \mathbf{X}$ and $Y \in \mathbf{Y}$. We show that $\mathcal{P}$ is blocked by $\mathbf{Z}'$ iff it is blocked by $\mathbf{Z}$ according to the definition in [GVP90]. We first show the if-part. Suppose there is a convergent valve² for variable W that is closed when conditioned on $\mathbf{Z}$, then the valve is still closed when conditioned on $\mathbf{Z}'$ unless the parents of W are in $\mathbf{Z}'$. However, the path $\mathcal{P}$ will be blocked in the latter case since the parents of W must have sequential/divergent valves. Suppose there is a sequential/divergent valve that is closed when conditioned on $\mathbf{Z}$ according to [GVP90], then W must be in $\mathbf{Z}'$ since it is functionally determined by $\mathbf{Z}$. Hence, the valve is also closed when conditioned on $\mathbf{Z}'$.

We next show the only-if part. Suppose a convergent valve for variable W is closed when conditioned on $\mathbf{Z}'$, then none of $\mathbf{Z}$ is a descendent of W since $\mathbf{Z}'$ is a superset of $\mathbf{Z}$. Suppose a sequential/divergent valve for variable W is closed when conditioned on $\mathbf{Z}'$, then W is functionally determined by $\mathbf{Z}$ by the construction of $\mathbf{Z}'$. Thus, the valve is closed in [GVP90]. $\square$

Proof of Theorem 1. By induction, it suffices to show that $\mathbf{X}$ and $\mathbf{Y}$ are D-separated by $\mathbf{Z}$

²See [Dar09, Ch. 4] for more details on convergent, divergent and sequential valves.

in $\langle G, \mathbf{W} \rangle$ iff they are D-separated by $\mathbf{Z}$ in $\langle G', \mathbf{W}' \rangle$, where G' is the result of functionally eliminating a single variable $T \in \mathbf{W}$ from G and $\mathbf{W}' = \mathbf{W} \setminus \{T\}$. We first show the contrapositive of the if-part. Suppose $\mathbf{X}$ and $\mathbf{Y}$ are not D-separated by $\mathbf{Z}$ in $\langle G, \mathbf{W} \rangle$, by the completeness of D-separation, there exists a parameterization $\mathcal{F}$ on G such that $(\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} | \mathbf{Z})_{\mathcal{F}}$. If we eliminate T from the CBN $\langle G, \mathcal{F} \rangle$, we obtain another CBN $\langle G', \mathcal{F}' \rangle$ where $\mathcal{F}'$ is the parameterization for G' . By Theorem 15, the marginal probabilities are preserved for the variables in G' , which include $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$. Hence, $(\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} | \mathbf{Z})_{\mathcal{F}'}$ and $\mathbf{X}$ and $\mathbf{Y}$ are not D-separated by $\mathbf{Z}$ in $\langle G', \mathbf{W}' \rangle$.

Next consider the contrapositive of the only-if part. Suppose $\mathbf{X}$ and $\mathbf{Y}$ are not D-separated by $\mathbf{Z}$ in $\langle G', \mathbf{W}' \rangle$, then there exists a parameterization $\mathcal{F}'$ of G' such that $(\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} | \mathbf{Z})_{\mathcal{F}'}$ by the completeness of D-separation. We construct a parameterization $\mathcal{F}$ for G such that $\mathcal{F}'$ is the parameterization of G' which results from eliminating T from the CBN $\langle G, \mathcal{F} \rangle$. This is sufficient to show that $\mathbf{X}$ and $\mathbf{Y}$ are not D-separated by $\mathbf{Z}$ in $\langle G, \mathbf{W} \rangle$ since the marginals are preserved by Theorem 15.

Construction Method Let $\mathbf{P}_T$ and $\mathbf{C}_T$ denote the parents and children of T in G . Our construction assumes that the cardinality of T is the number of instantiations for its parents $\mathbf{P}_T$. That is, there is a one-to-one correspondence between the states of T and the instantiations of $\mathbf{P}_T$, and we use $\alpha(t)$ to denote the instantiation $\mathbf{p}_T$ corresponding to state t . The functional CPT for T is assigned as $f_T(t | \mathbf{p}_T) = 1$ if $\alpha(t) = \mathbf{p}_T$ and $f_T(t | \mathbf{p}_T) = 0$ otherwise for each instantiation $\mathbf{p}_T$ of $\mathbf{P}_T$. Now consider each child $C \in \mathbf{C}_T$ that has parents $\mathbf{P}_C$ (excluding T) and T in G . It immediately follows from Definition 6 that C has parents $\mathbf{P}_T \cup \mathbf{P}_C$ in G' . We next construct the CPT f_C in $\mathcal{F}$ based on its CPT f'_C in $\mathcal{F}'$. Consider each parent instantiation $(t, \mathbf{p}_C)$ where t is a state of T and $\mathbf{p}_C$ is an instantiation of $\mathbf{P}_C$. If $\alpha(t)$ is consistent with $\mathbf{p}_C$, assign $f_C(c | t, \mathbf{p}_C) = f'_C(c | \alpha(t), \mathbf{p}_C)$ for each state c .³ Otherwise, assign any functional distribution for $f_C(c | t, \mathbf{p}_C)$. The construction ensures that the constructed CPT f_T for T is functional, and that the functional dependencies among other variables are

³For clarity, we use the notation $|$ to separate a variable and its parents in a CPT.

preserved. In particular, for each child C of T , the constructed CPT f_C is functional iff f'_C is functional. This construction method will be reused later in other proofs.

We now just need to show that CBN $\langle G', \mathcal{F}' \rangle$ is the result of eliminating T from the (constructed) CBN $\langle G, \mathcal{F} \rangle$. In particular, we need to check that the CPT for each child $C \in \mathbf{C}_T$ in $\mathcal{F}'$ is correctly computed from the constructed CPTs in $\mathcal{F}$. For each instantiation $(\mathbf{p}_T, \mathbf{p}_C)$ and state c of C ,

$$f'_C(c|\mathbf{p}_T, \mathbf{p}_C) = f_C(c|t^*, \mathbf{p}_C) = f_T(t^*|\mathbf{p}_T)f_C(c|t^*, \mathbf{p}_C) = \sum_t f_T(t|\mathbf{p}_T)f_C(c|t, \mathbf{p}_C)$$

where t^* is the state of T such that $\alpha(t^*) = \mathbf{p}_T$. □

Proof of Theorem 2. We prove the theorem by induction. It suffices to show the following statement: for each causal graph G with observed variables $\mathbf{V}$ and functional variables $\mathbf{W}$, the causal effect $\text{Pr}_{\mathbf{x}}(\mathbf{Y})$ is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ iff it is identifiable wrt $\langle G', \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}'\} \rangle$ where G' is the result of functionally eliminating some hidden functional variable $T \in \mathbf{W}$ and $\mathbf{W}' = \mathbf{W} \setminus \{T\}$.

We first show the contrapositive of the if-part. Suppose $\text{Pr}_{\mathbf{x}}(\mathbf{Y})$ is not identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$, there exist two CBNs $\langle G, \mathcal{F}_1 \rangle$ and $\langle G, \mathcal{F}_2 \rangle$ which induce distributions Pr_1, Pr_2 such that $\text{Pr}_1(\mathbf{V}) = \text{Pr}_2(\mathbf{V})$ but $\text{Pr}_{1\mathbf{x}}(\mathbf{Y}) \neq \text{Pr}_{2\mathbf{x}}(\mathbf{Y})$. Let $\langle G', \mathcal{F}'_1 \rangle$ and $\langle G', \mathcal{F}'_2 \rangle$ be the results of eliminating $T \notin \mathbf{V}$ from $\langle G, \mathcal{F}_1 \rangle$ and $\langle G, \mathcal{F}_2 \rangle$, the two CBNs attain the same marginal distribution on $\mathbf{V}$ but different causal effects by Theorems 15 and 16. Hence, $\text{Pr}_{\mathbf{x}}(\mathbf{Y})$ is not identifiable wrt $\langle G', \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}'\} \rangle$ either.

We next show the contrapositive of the only-if part. Suppose $\text{Pr}_{\mathbf{x}}(\mathbf{Y})$ is not identifiable wrt $\langle G', \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}'\} \rangle$, there exist two CBNs $\langle G', \mathcal{F}'_1 \rangle$ and $\langle G', \mathcal{F}'_2 \rangle$ which induce distributions $\text{Pr}'_1, \text{Pr}'_2$ such that $\text{Pr}'_1(\mathbf{V}) = \text{Pr}'_2(\mathbf{V})$ but $\text{Pr}'_{1\mathbf{x}}(\mathbf{Y}) \neq \text{Pr}'_{2\mathbf{x}}(\mathbf{Y})$. We can obtain $\langle G, \mathcal{F}_1 \rangle$ and $\langle G, \mathcal{F}_2 \rangle$ by considering again the construction method in Theorem 1 where we assign a one-to-one mapping for T and adopt the CPTs from $\mathcal{F}'_1$ and $\mathcal{F}'_2$ for the children of T . This way, $\langle G', \mathcal{F}'_1 \rangle$ and $\langle G', \mathcal{F}'_2 \rangle$ become the results of eliminating T from the constructed $\langle G, \mathcal{F}_1 \rangle$

and $\langle G, \mathcal{F}_2 \rangle$. Since $T \notin \mathbf{V}$, $\Pr_1(\mathbf{V}) = \Pr'_1(\mathbf{V}) = \Pr'_2(\mathbf{V}) = \Pr_2(\mathbf{V})$ by Theorem 15, and $\Pr_{1\mathbf{x}}(\mathbf{Y}) = \Pr'_{1\mathbf{x}}(\mathbf{Y}) \neq \Pr'_{2\mathbf{x}}(\mathbf{Y}) = \Pr_{2\mathbf{x}}(\mathbf{Y})$ by Theorem 16. Hence, $\Pr_{\mathbf{x}}(\mathbf{Y})$ is not identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ either. $\square$

Proof of Theorem 3. Since we only functionally eliminate variables that have observed parents, it is guaranteed that each $Z \in \mathbf{Z}$ has observed parents when it is eliminated. By induction, it suffices to show that $\Pr_{\mathbf{x}}(\mathbf{Y})$ is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ iff it is identifiable wrt $\langle G', \mathbf{V}', \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}'\} \rangle$ where G' is the result of eliminating a single functional variable $Z \in \mathbf{W}$ with observed parents from G , $\mathbf{V}' = \mathbf{V} \setminus \{Z\}$, and $\mathbf{W}' = \mathbf{W} \setminus \{Z\}$.

We first show the contrapositive of the if-part. Suppose $\Pr_{\mathbf{x}}(\mathbf{Y})$ is not identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$, there exist two CBNs $\langle G, \mathcal{F}_1 \rangle$ and $\langle G, \mathcal{F}_2 \rangle$ which induce distributions $\Pr_1, \Pr_2$ where $\Pr_1(\mathbf{V}) = \Pr_2(\mathbf{V})$ but $\Pr_{1\mathbf{x}}(\mathbf{Y}) \neq \Pr_{2\mathbf{x}}(\mathbf{Y})$. Let $\langle G', \mathcal{F}'_1 \rangle$ and $\langle G', \mathcal{F}'_2 \rangle$ be the results of eliminating Z from $\langle G, \mathcal{F}_1 \rangle$ and $\langle G, \mathcal{F}_2 \rangle$, the two CBNs induce the same marginal distribution $\Pr'_1(\mathbf{V}') = \Pr'_2(\mathbf{V}')$ by Theorem 15 but different causal effects $\Pr'_{1\mathbf{x}}(\mathbf{Y}) \neq \Pr'_{2\mathbf{x}}(\mathbf{Y})$ by Theorem 16. Hence, $\Pr_{\mathbf{x}}(\mathbf{Y})$ is unidentifiable wrt $\langle G', \mathbf{V}', \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}'\} \rangle$.

We now consider the contrapositive of the only-if part. Suppose $\Pr_{\mathbf{x}}(\mathbf{Y})$ is not identifiable wrt $\langle G', \mathbf{V}', \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}'\} \rangle$, then there exist two CBNs $\langle G', \mathcal{F}'_1 \rangle$ and $\langle G', \mathcal{F}'_2 \rangle$ which induce distributions $\Pr'_1, \Pr'_2$ such that $\Pr'_1(\mathbf{V}') = \Pr'_2(\mathbf{V}')$ but $\Pr'_{1\mathbf{x}}(\mathbf{Y}) \neq \Pr'_{2\mathbf{x}}(\mathbf{Y})$. We again consider the construction method from the proof of Theorem 1 which produces two CBNs $\langle G, \mathcal{F}_1 \rangle$ and $\langle G, \mathcal{F}_2 \rangle$. Since $\langle G', \mathcal{F}'_1 \rangle$ and $\langle G', \mathcal{F}'_2 \rangle$ are the results of eliminating Z from $\langle G, \mathcal{F}_1 \rangle$ and $\langle G, \mathcal{F}_2 \rangle$, the constructed CBNs produce different values for causal effects $\Pr_{1\mathbf{x}}(\mathbf{Y}) = \Pr'_{1\mathbf{x}}(\mathbf{Y}) \neq \Pr'_{2\mathbf{x}}(\mathbf{Y}) = \Pr_{2\mathbf{x}}(\mathbf{Y})$ by Theorem 16. We are left to show that $\langle G, \mathcal{F}_1 \rangle$ and $\langle G, \mathcal{F}_2 \rangle$ induce the same distribution over variables $\mathbf{V} = \mathbf{V}' \cup \{Z\}$. Consider any instantiation $(\mathbf{v}', z)$ of $\mathbf{V}$. Since $\mathcal{F}_1$ and $\mathcal{F}_2$ assign the same one-to-one mapping α between Z and its parents in G , it is guaranteed that the probabilities $\Pr_1(\mathbf{v}', z) = \Pr_2(\mathbf{v}', z) = 0$ except for the single state z^* where $\alpha(z^*) = \mathbf{p}$ where $\mathbf{p}$ is the parent instantiation of Z consistent with $\mathbf{v}'$. By construction, $\Pr_1(\mathbf{v}', z^*, \mathbf{u}) = \Pr'_1(\mathbf{v}', \mathbf{u})$ for every instantia-



Figure B.1: Variables X, Y_1, Z, Y_2 are observed. Variable Z is functional.

tion $\mathbf{u}$ of hidden variables $\mathbf{U}$. Similarly, $\Pr_2(\mathbf{v}', z^*, \mathbf{u}) = \Pr'_2(\mathbf{v}', \mathbf{u})$ for every instantiation $(\mathbf{v}', z, \mathbf{u})$. It then follows that $\Pr_1(\mathbf{v}', z^*) = \sum_{\mathbf{u}} \Pr_1(\mathbf{v}', z^*, \mathbf{u}) = \sum_{\mathbf{u}} \Pr'_1(\mathbf{v}', \mathbf{u}) = \Pr'_1(\mathbf{v}') = \Pr'_2(\mathbf{v}') = \sum_{\mathbf{u}} \Pr'_2(\mathbf{v}', \mathbf{u}) = \sum_{\mathbf{u}} \Pr_2(\mathbf{v}', z^*, \mathbf{u}) = \Pr_2(\mathbf{v}', z^*)$. This indicates that $\Pr_{\mathbf{x}}(\mathbf{Y})$ is not identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ either. $\square$

Proposition 13. *Theorem 3 may not hold if an observed functional variable in $\mathbf{Z}$ has a hidden parent.*

Proof. Consider the causal graph G in Figure B.1(a) in which Z is a functional variable with a hidden parent U_2 . Assume the positivity constraint $\Pr(X, Y_1, Y_2) > 0$, which ensures the separability property. Applying the ID algorithm to G yields the following formula for computing the causal effect of X on Y_1 and Y_2 :

$$\Pr_x(y_1, y_2) = \sum_z \Pr(y_2|x, z) \Pr(y_1|z) \Pr(z) = \sum_{\{z | \Pr(z) > 0\}} \Pr(y_2|x, z) \Pr(y_1|z) \Pr(z)$$

Since $(X \perp\!\!\!\perp Z)$ in the graph, $\Pr(x, z) > 0$ whenever $\Pr(z) > 0$. Hence, all conditional probabilities in the formula are well-defined and the causal effect is identifiable under the positivity constraint. However, if we eliminate the functional variable Z from G , the causal effect becomes unidentifiable on the resulting graph shown in Figure B.1(b). $\square$

We start with a lemma, which will be used for the proof of Theorem 4.

Lemma 2. *Let G be a causal graph, $\mathbf{V}$ be its observed variables and $\mathbf{W}$ be its functional variables. Let Z be a non-descendant of $\mathbf{W}$ that has at least one hidden parent, then a causal*

effect is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ iff it is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W} \cup \{Z\}\} \rangle$.

Proof. Let $\mathbf{W}'$ denote the set $\mathbf{W} \cup \{Z\}$. The only-if part holds immediately by the fact that every distribution that can be possibly induced from $\langle G, \mathbf{W}' \rangle$ can also be induced from $\langle G, \mathbf{W} \rangle$. We next show the contrapositive of the if part. Suppose a causal effect is not identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$, there exist two CBNs $\langle G, \mathcal{F}_1 \rangle$ and $\langle G, \mathcal{F}_2 \rangle$ which induce distributions Pr_1, Pr_2 such that $\text{Pr}_1(\mathbf{V}) = \text{Pr}_2(\mathbf{V})$ but $\text{Pr}_{1\mathbf{x}}(\mathbf{Y}) \neq \text{Pr}_{2\mathbf{x}}(\mathbf{Y})$. We next construct $\langle G, \mathcal{F}_1'' \rangle$ and $\langle G, \mathcal{F}_2'' \rangle$ which constitute an example of unidentifiability wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}' \rangle$, which functional CPTs for Z .

WLG, we show the construction of $\langle G, \mathcal{F}_1'' \rangle$ from $\langle G, \mathcal{F}_1 \rangle$ which involves two steps (the construction of $\langle G, \mathcal{F}_2'' \rangle$ from $\langle G, \mathcal{F}_2 \rangle$ will follow a same procedure). Let $\mathbf{P}$ be the parents of Z . The first step constructs a CBN $\langle G', \mathcal{F}_1' \rangle$ based on the known method that transforms any (non-functional) CPT into a functional CPT. This is done by adding an auxiliary hidden root parent U for Z whose states correspond to the possible functions between $\mathbf{P}$ and Z . The CPTs for U and Z are assigned accordingly such that $f_Z = \sum_U f'_U f'_Z$ where f'_U and f'_Z are the constructed CPTs in $\mathcal{F}_1'$.⁴ It follows that $\mathcal{F}_1$ and $\mathcal{F}_1'$ induce the same distribution over $\mathbf{V}$ since $\mathcal{F}_1 = \sum_U \mathcal{F}_1'$. The causal effect is also preserved since summing out U is independent of other CPTs in the mutilated CBN for $\langle G', \mathcal{F}_1' \rangle$.

Our second step involves converting the CBN $\langle G', \mathcal{F}_1' \rangle$ (constructed from the first step) into the CBN $\langle G, \mathcal{F}_1'' \rangle$ over the original graph G . Let $T \in \mathbf{P}$ be the hidden parent of W in G . We merge the auxiliary parent U and T into a new variable T' and substitute it for T in G , i.e., T' has the same parents and children as T . T' is constructed as the Cartesian product of U and T : each state of T' is represented as a pair (u, t) where u is a state of U and t is a state of T . We are ready to assign new CPTs for T' and its children. For each parent instantiation $\mathbf{p}$ of $\mathbf{P}$ and each state (u, t) of T' , we assign the CPT for T' in

⁴Each state u of U corresponds to a function γ_u where $\gamma_u(\mathbf{p})$ is mapped to some state of Z for each instantiation $\mathbf{p}$. The variable U thus has $|Z|^{|P|}$ states since there are total of $|Z|^{|P|}$ possible functions from $\mathbf{P}$ to Z . For each instantiation $(z, u, \mathbf{p})$, the functional CPT for Z is defined as $f'_Z(z|u, \mathbf{p}) = 1$ if $z = \gamma_u(\mathbf{p})$, and $f'_Z(z|u, \mathbf{p}) = 0$ otherwise. The CPT for U is assigned as $f'_U(u) = \prod_{\mathbf{p} \in \mathbf{P}} f_Z(\gamma_u(\mathbf{p})|\mathbf{p})$.

$\mathcal{F}''$ as $f_{T'}''((u, t)|\mathbf{p}) = f_U'(u)f_T'(t|\mathbf{p})$. Next consider each child C of T' that has parents $\mathbf{P}_C$ (excluding T'). For each instantiation $\mathbf{p}_C$ of $\mathbf{P}_C$ and each state (u, t) of T' , we assign the CPT for C in $\mathcal{F}''$ as $f_C''(c|\mathbf{p}_C, (u, t)) = f_C'(c|\mathbf{p}_C)$ for each state c . Note that f_C'' is functional iff f_C' is functional. Hence, the CPTs for $\mathbf{W}$ are all functional in $\mathcal{F}''$.

We need to show that $\langle G, \mathcal{F}_1'' \rangle$ preserves the distribution on $\mathbf{V}$ and the causal effect from $\langle G', \mathcal{F}_1' \rangle$. Let $\mathbf{U}''$ be the hidden variables in $\langle G, \mathcal{F}_1'' \rangle$ and $\mathbf{U}'$ be the hidden variables in $\langle G', \mathcal{F}_1' \rangle$. The distribution on $\mathbf{V}$ is preserved since there is a one-to-one correspondence between each instantiation $(\mathbf{v}, \mathbf{u}'')$ in $\langle G, \mathcal{F}_1'' \rangle$ and each instantiation $(\mathbf{v}, \mathbf{u}')$ in $\langle G', \mathcal{F}_1' \rangle$ where the two instantiations agree on $\mathbf{v}$ and are assigned with the same probability, i.e., $\text{Pr}_1''(\mathbf{v}, \mathbf{u}'') = \text{Pr}_1'(\mathbf{v}, \mathbf{u}')$. Hence, $\text{Pr}_1''(\mathbf{v}) = \sum_{\mathbf{u}''} \text{Pr}_1''(\mathbf{v}, \mathbf{u}'') = \sum_{\mathbf{u}'} \text{Pr}_1'(\mathbf{v}, \mathbf{u}') = \text{Pr}_1'(\mathbf{v})$ for every instantiation $\mathbf{v}$. The preservation of causal effect can be shown similarly but on the mutilated CBNs. Thus, $\langle G, \mathcal{F}_1'' \rangle$ preserves both the distribution on $\mathbf{V}$ and the causal effect from $\langle G, \mathcal{F}_1 \rangle$. Similarly, we can construct $\langle G, \mathcal{F}_2'' \rangle$ which preserves the distribution on $\mathbf{V}$ and causal effect from $\langle G, \mathcal{F}_2 \rangle$. The two CBNs $\langle G, \mathcal{F}_1'' \rangle$ and $\langle G, \mathcal{F}_2'' \rangle$ constitute an example of unidentifiability wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}'\} \rangle$. $\square$

Proof of Theorem 4. We prove the theorem by induction. We first order all the functional variables in a bottom-up order. Let W^i denote the i^{th} functional variable in the order and $\mathbf{W}^{(i)}$ denote the functional variables that are ordered before W^i (including W^i). It follows that we can go over each W^i in the order and show that a causal effect is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}^{(i-1)}\} \rangle$ iff it is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}^{(i)}\} \rangle$ by Lemma 2. Since identifiability wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}^{(0)}\} \rangle$ is equivalent to identifiability wrt $\langle G, \mathbf{V}, \mathcal{C}_{\mathbf{V}} \rangle$, we conclude that the causal effect is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ iff it is identifiable wrt $\langle G, \mathbf{V}, \mathcal{C}_{\mathbf{V}} \rangle$. $\square$

The proof of the Theorem 5 is organized as follows. We start with a lemma (Lemma 3) that allows us to modify the CPT of a variable when the marginal probability over its parents contain zero entries. We then show a main lemma (Lemma 4) that allows us to reduce constrained identifiability to classical identifiability when all functional variables are

observed or has a hidden parent. We finally prove the theorem based on the main lemma and previous theorems.

Lemma 3. *Consider two CBNs that have a same causal graph and induce the distributions $\Pr_1$ and $\Pr_2$. Suppose the CPTs of the two CBNs only differ by $f_X(X, \mathbf{P})$. Then $\Pr_1(\mathbf{p}) = 0$ iff $\Pr_2(\mathbf{p}) = 0$ for all instantiations $\mathbf{p}$ of $\mathbf{P}$.*

Proof. Let f_Y and g_Y denote the CPT for Y in the first and second CBN. Let $\mathcal{An}(\mathbf{P})$ be the ancestors of variables in $\mathbf{P}$ (including $\mathbf{P}$). If we eliminate all variables other than $\mathcal{An}(\mathbf{P})$, then we obtain the factor set $\mathcal{F}_1 = \prod_{Y \in \mathcal{An}(\mathbf{P})} f_Y$ for the first CBN and $\mathcal{F}_2 = \prod_{Y \in \mathcal{An}(\mathbf{P})} g_Y$ for the second CBN. Since all CPTs are the same for variables in $\mathcal{An}(\mathbf{P})$, it is guaranteed that $\mathcal{F}_1 = \mathcal{F}_2$. If we further eliminate variables other than $\mathbf{P}$ from $\mathcal{F}_1$ and $\mathcal{F}_2$, we obtain the marginal distributions $\Pr_1(\mathbf{P}) = \sum_{\mathbf{P}} \mathcal{F}_1$ and $\Pr_2(\mathbf{P}) = \sum_{\mathbf{P}} \mathcal{F}_2$, where $\sum_{\mathbf{P}}$ denotes the projection operation that sums out variables other than $\mathbf{P}$ from a factor. Hence, $\Pr_1(\mathbf{P}) = \Pr_2(\mathbf{P})$ which concludes the proof. $\square$

Lemma 4. *If a causal effect is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ but is not identifiable wrt $\langle G, \mathbf{V}, \mathcal{C}_{\mathbf{V}} \rangle$, then there must exist at least one functional variable that is hidden and whose parents are all observed.*

Proof. The lemma is the same as saying that if every functional variable is observed or having a hidden parent, then constrained identifiability under functional dependencies is equivalent to classical identifiability. We go over each functional variable $W_i \in \mathbf{W}$ in a bottom-up order Π and show the following inductive statement: a causal effect $\Pr_x(\mathbf{Y})$ is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}^{(i)}\} \rangle$ iff it is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}^{(i-1)}\} \rangle$, where $\mathbf{W}^{(i)}$ is a subset of variables in $\mathbf{W}$ that are ordered before W_i (and including W_i) in Π . Note that $\mathbf{W}^{(0)} = \emptyset$ and identifiability wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}^{(0)}\} \rangle$ collapses into identifiability wrt $\langle G, \mathbf{V}, \mathcal{C}_{\mathbf{V}} \rangle$.

The if-part follows from the definition of constrained identifiability. We now consider the contrapositive of the only-if part. Let Z be the functional variable in $\mathbf{W}$ that is considered in

the current inductive step. Let $\langle G, \mathcal{F}_1 \rangle$ and $\langle G, \mathcal{F}_2 \rangle$ be the two CBNs inducing distributions $\Pr_1$ and $\Pr_2$ that constitute unidentifiability, i.e., $\Pr_1(\mathbf{V}) = \Pr_2(\mathbf{V})$ and $\Pr_{1x}(\mathbf{Y}) \neq \Pr_{2x}(\mathbf{Y})$. Our goal is to construct two CBNs $\langle G, \mathcal{F}_1' \rangle$ and $\langle G, \mathcal{F}_2' \rangle$, which induce distributions $\Pr_1', \Pr_2'$ and contain functional CPTs for Z , such that $\Pr_1'(\mathbf{V}) = \Pr_2'(\mathbf{V})$ and $\Pr_{1x}'(\mathbf{Y}) \neq \Pr_{2x}'(\mathbf{Y})$. Suppose Z has a hidden parent, we directly employ Lemma 2 to construct the two CBNs. We next consider the case when Z is observed and has observed parents. By default, we use f_Z to g_Z to denote the CPT for Z in $\mathcal{F}_1$ and $\mathcal{F}_2$.

The following three steps are considered to construct an instance of unidentifiability.

First Step: we construct $\langle G, \mathcal{F}_1' \rangle$ and $\langle G, \mathcal{F}_2' \rangle$ by modifying the CPTs for Z . Let $\mathbf{P}$ be the parents of Z in G . For each instantiation $\mathbf{p}$ of $\mathbf{P}$ where $\Pr_1(\mathbf{p}) = \Pr_2(\mathbf{p}) = 0$, we modify the entries $f_Z'(Z|\mathbf{p})$ and $g_Z'(Z|\mathbf{p})$ for the CPTs f_Z' and g_Z' in $\mathcal{F}_1'$ and $\mathcal{F}_2'$ as follows. Since $\Pr_{1x}(\mathbf{Y}) \neq \Pr_{2x}(\mathbf{Y})$, there exists an instantiation $\mathbf{y}$ such that $\Pr_{1x}(\mathbf{y}) \neq \Pr_{2x}(\mathbf{y})$. WLG, assume $\Pr_{1x}(\mathbf{y}) > \Pr_{2x}(\mathbf{y})$. Since $\Pr_{1x}(\mathbf{y})$ is computed as the marginal probability of $\mathbf{y}$ in the mutilated CBN for $\langle G, \mathcal{F}_1 \rangle$, it can be expressed in the form of *network polynomial* as shown in [Dar03, Dar09]. If we treat the CPT entries of $f_Z(Z|\mathbf{p})$ as unknown, then we can write $\Pr_{1x}(\mathbf{y})$ as follows

$$\Pr_{1x}(\mathbf{y}) = \alpha_0 + \alpha_1 f_Z(z_1|\mathbf{p}) + \cdots + \alpha_k f_Z(z_k|\mathbf{p})$$

where $\alpha_0, \alpha_1, \dots, \alpha_k$ are constants and $z_1, \dots, z_k$ are the states of variable Z . Similarly, we can write $\Pr_{2x}(\mathbf{y})$ as follows

$$\Pr_{2x}(\mathbf{y}) = \beta_0 + \beta_1 g_Z(z_1|\mathbf{p}) + \cdots + \beta_k g_Z(z_k|\mathbf{p})$$

Let α_i be the *maximum* value among $\alpha_1, \dots, \alpha_k$ and β_j be the *minimum* value among $\beta_1, \dots, \beta_k$, our construction method assigns $f_Z'(z_i|\mathbf{p}) = 1$ and $g_Z'(z_j|\mathbf{p}) = 1$. By construction, it is guaranteed that $\Pr_{1x}'(\mathbf{y}) - \Pr_{2x}'(\mathbf{y}) \geq \Pr_{1x}(\mathbf{y}) - \Pr_{2x}(\mathbf{y}) > 0$ where $\Pr_{1x}'(\mathbf{y})$, $\Pr_{2x}'(\mathbf{y})$ denote the causal effects under the updated CPTs $f_Z'(Z|\mathbf{p})$ and $g_Z'(Z|\mathbf{p})$. We repeat the above procedure for all such $\mathbf{p}$ where $\Pr_1(\mathbf{p}) = 0$, which yields the new CBNs $\langle G, \mathcal{F}_1' \rangle$ and

$\langle G, \mathcal{F}'_2 \rangle$ in which $f'_Z(Z|\mathbf{p})$ and $g'_Z(Z|\mathbf{p})$ are functional whenever $\Pr_1(\mathbf{p}) = 0$. We next show that $\mathcal{F}'_1$ and $\mathcal{F}'_2$ (with the updated f'_Z and g'_Z) constitute an example of unidentifiability. $\Pr'_{1x}(\mathbf{Y}) \neq \Pr'_{2x}(\mathbf{Y})$ since $\Pr'_{1x}(\mathbf{y}) > \Pr'_{2x}(\mathbf{y})$ for the particular instantiation $\mathbf{y}$. We are left to show that the distributions $\Pr'_1$ and $\Pr'_2$ induced by $\langle G, \mathcal{F}'_1 \rangle$ and $\langle G, \mathcal{F}'_2 \rangle$ are the same over the observed variables $\mathbf{V}$. Consider each instantiation $\mathbf{v}$ of $\mathbf{V}$ and $\mathbf{p}$ of $\mathbf{P}$ where $\mathbf{p}$ is consistent with $\mathbf{v}$. If $\Pr_1(\mathbf{p}) = 0$, then $\Pr'_1(\mathbf{p}) = \Pr'_2(\mathbf{p}) = 0$ by Lemma 3 and thus $\Pr'_1(\mathbf{v}) = \Pr'_2(\mathbf{v}) = 0$. Otherwise, $\Pr'_1(\mathbf{v}) = \Pr_1(\mathbf{v}) = \Pr_2(\mathbf{v}) = \Pr'_2(\mathbf{v})$ since none of the CPT entries consistent with $\mathbf{v}$ were modified.

Second Step: We construct $\langle G'', \mathcal{F}''_1 \rangle$, $\langle G'', \mathcal{F}''_2 \rangle$ from $\langle G, \mathcal{F}'_1 \rangle$, $\langle G, \mathcal{F}'_2 \rangle$ by introducing an auxiliary root parent for Z and assigning a functional CPT for Z . We add a root variable, denoted R , to be an auxiliary parent of Z which specifies all possible functions from $\mathbf{P}^*$ to Z where $\mathbf{P}^*$ contains all instantiations of $\mathbf{p}^*$ where $\Pr'_1(\mathbf{p}^*) = \Pr'_2(\mathbf{p}^*) > 0$. Each state r of R corresponds to a function φ_r where $\varphi_r(\mathbf{p}^*)$ is mapped to some state of Z for each instantiation $\mathbf{p}^*$. The variable R thus has $|Z|^{|P^*|}$ states since there are total of $|Z|^{|P^*|}$ possible functions from $\mathbf{P}^*$ to Z . For each instantiation $(z, r, \mathbf{p})$, if $\mathbf{p} \in \mathbf{P}^*$, we define $f''_Z(z|r, \mathbf{p}) = 1$ if $z = \varphi_r(\mathbf{p})$, and $f''_Z(z|r, \mathbf{p}) = 0$ otherwise. If $\mathbf{p} \notin \mathbf{P}^*$, we define $f''_Z(z|r, \mathbf{p}) = f'_Z(z|\mathbf{p})$. The CPT for R is assigned as $f''_R(r) = \prod_{\mathbf{p} \in \mathbf{P}^*} f'_Z(\varphi_r(\mathbf{p})|\mathbf{p})$. Moreover, we remove all the states r of R where $f''_R(r) = 0$, which ensures $f''_R(r) > 0$ for all remaining r . We assign the CPT g''_Z in $\mathcal{F}''_2$ in a similar way. Note that $f''_R = g''_R$ since $f'_Z(\varphi_r(\mathbf{p})|\mathbf{p}) = \Pr'_1(\varphi_r(\mathbf{p})|\mathbf{p}) = \Pr'_2(\varphi_r(\mathbf{p})|\mathbf{p}) = g'_Z(\varphi_r(\mathbf{p})|\mathbf{p})$ for each $\mathbf{p} \in \mathbf{P}^*$ (where $\Pr'_1(\mathbf{p}) = \Pr'_2(\mathbf{p}) > 0$).

We next show that $\langle G'', \mathcal{F}''_1 \rangle$ and $\langle G'', \mathcal{F}''_2 \rangle$ constitute unidentifiability. One key observation is that $f''_T(t|\mathbf{p}) = \sum_r f''_R(r) f''_T(t|\mathbf{p}, r)$ and $g''_T(t|\mathbf{p}) = \sum_r g''_R(r) g''_T(t|\mathbf{p}, r)$. Consider each instantiation $(\mathbf{v}, r)$ which contains the instantiation $\mathbf{p}$ of $\mathbf{P}$ and the state z of Z . Suppose $\Pr'_1(\mathbf{p}) = 0$, then $\Pr''_1(\mathbf{p}) = 0$ since the marginal over $\mathbf{V}$ is preserved in $\Pr''_1$. Hence, $\Pr''_1(\mathbf{v}, r) = \Pr''_2(\mathbf{v}, r) = 0$. Suppose $\Pr'_1(\mathbf{p}) \neq 0$, then $\Pr''_1(\mathbf{v}, r) = (\prod_{V \in \mathbf{V} \setminus \{Z\}} f'_V) \cdot f''_R(r)$ when $z = \varphi_r(\mathbf{p})$, and $\Pr''_1(\mathbf{v}, r) = 0$ otherwise. Similarly, $\Pr''_2(\mathbf{v}, r) = (\prod_{V \in \mathbf{V} \setminus \{Z\}} g'_V) \cdot g''_R(r)$ when $z = \varphi_r(\mathbf{p})$, and $\Pr''_2(\mathbf{v}, r) = 0$ otherwise. In both cases, $\Pr''_1(\mathbf{v}, r) = \Pr''_2(\mathbf{v}, r)$ since $\mathcal{F}''_1$ and

$\mathcal{F}_2''$ assign a same function φ_r for each state r of R , and $f'_V = g'_V$ for all $V \in \mathbf{V} \setminus \{Z\}$. To see $\Pr''_{1x}(\mathbf{Y}) = \Pr'_{1x}(\mathbf{Y})$ and $\Pr''_{2x}(\mathbf{Y}) = \Pr'_{2x}(\mathbf{Y})$, note that summing-out R from $\Pr''_{1x}(\mathbf{V}, R)$ and $\Pr''_{2x}(\mathbf{V}, R)$ yields $\Pr'_{1x}(\mathbf{V})$ and $\Pr'_{2x}(\mathbf{V})$.

Third Step: we construct $\langle G, \mathcal{F}_1''' \rangle, \langle G, \mathcal{F}_2''' \rangle$ from $\langle G'', \mathcal{F}_1'' \rangle, \langle G'', \mathcal{F}_2'' \rangle$ by merging the auxiliary root variable R with an observed parent T of Z . We merge R and T into a new node T' and substitute it for T in G , i.e., T' has the same parents and children as T in G . Specifically, T' is constructed as the Cartesian product of R and T : each state of T' can be represented as (r, t) where r is a state of R and t is a state of T . We then assign the CPT $f'''_{T'}(T'|\mathbf{P}_{T'})$ in $\mathcal{F}_1'''$ as follows. For each parent instantiation $\mathbf{p}_{T'}$ and each state (r, t) of T' , $f'''_{T'}((r, t)|\mathbf{p}_{T'}) = f''_R(r)f''_T(t|\mathbf{p}_{T'})$. For each child C of T' that has parents $\mathbf{P}_C$ (excluding T'), the CPT for C in $\mathcal{F}_1'''$ is assigned as $f'''_C(c|\mathbf{p}_C, (r, t)) = f''_C(c|\mathbf{p}_C, t)$. Similarly, we assign the CPTs $g'''_{T'}$ and g'''_C in $\mathcal{F}_2'''$. The distributions over observed variables are preserved since there is an one-to-one correspondence between the instantiations over $\mathbf{V} \cup \{R\}$ in $\langle G'', \mathcal{F}_1'' \rangle$ and the instantiations over $\mathbf{V}$ in $\langle G, \mathcal{F}_1''' \rangle$. We next consider the causal effect. Suppose T is neither a treatment nor an outcome variable, then the merging does not affect the causal effect by the one-to-one correspondence between instantiations and $\Pr'''_{1x}(\mathbf{Y}) = \Pr''_{1x}(\mathbf{Y}) \neq \Pr''_{2x}(\mathbf{Y}) = \Pr'''_{2x}(\mathbf{Y})$. Suppose T is an outcome variable, since $\Pr''_{1x}(\mathbf{Y}) \neq \Pr''_{2x}(\mathbf{Y})$, there exists an instantiation $(\mathbf{y}', r, t)$ where $\mathbf{Y}' = \mathbf{Y} \setminus \{T\}$ such that $\Pr''_{1x}(\mathbf{y}', r, t) \neq \Pr''_{2x}(\mathbf{y}', r, t)$. This implies $\Pr'''_{1x}(\mathbf{y}', (r, t)) \neq \Pr'''_{2x}(\mathbf{y}', (r, t))$ for the particular instantiation $\mathbf{y}'$ and the state (r, t) of T' . Suppose T is the treatment variable X , since $\Pr''_{1x}(\mathbf{Y}) \neq \Pr''_{2x}(\mathbf{Y})$, there exists an instantiation $(\mathbf{y}, r)$ such that $\Pr''_{1x}(\mathbf{y}, r) \neq \Pr''_{2x}(\mathbf{y}, r)$. Moreover, $\Pr''_{1x}(r) = \Pr''_1(r) = \Pr''_2(r) = \Pr''_{2x}(r) > 0$ (otherwise, $\Pr''_{1x}(\mathbf{y}, r) = \Pr''_{2x}(\mathbf{y}, r) = 0$). This implies $\Pr''_{1x}(\mathbf{y}|r) \neq \Pr''_{2x}(\mathbf{y}|r)$. Since R is a root in the mutilated CBN, $\Pr''_{1(xr)}(\mathbf{y}) = \Pr''_{1x}(\mathbf{y}|r) \neq \Pr''_{2x}(\mathbf{y}|r) = \Pr''_{2(xr)}(\mathbf{y})$. We now consider the treatment $do(T' = (r, x))$ on G instead of the treatment $do(R = r, X = x)$ on G'' . We have $\Pr'''_{1((r, x))}(\mathbf{y}) = \Pr'''_{1(r, x)}(\mathbf{y}) \neq \Pr'''_{2(r, x)}(\mathbf{y}) = \Pr'''_{2((r, x))}(\mathbf{y})$ for the particular state (r, x) of T' . Moreover, $\Pr'''_1((r, x)) = \Pr''_1(r, x) = \Pr''_1(r)\Pr''_1(x) > 0$ by the positivity assumption of $\Pr''_1(x)$. Thus, the positivity still holds for $\Pr'''_1$ and similarly for $\Pr'''_2$. $\square$

Proof of Theorem 5. Let $\mathbf{H} = \mathbf{V}' \setminus \mathbf{V}$ be the set of hidden functional variables that are functionally determined by $\mathbf{V}$. By Theorem 2, the causal effect is identifiable wrt $\langle G, \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ iff it is identifiable wrt $\langle G', \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W} \setminus \mathbf{H}\} \rangle$, where G' is the result of functionally eliminating $\mathbf{H}$ from G . By construction, every variable in $\mathbf{H}$ has parents in $\mathbf{V}'$. Hence, by Theorem 3, it is identifiable wrt $\langle G', \mathbf{V}, \{\mathcal{C}_{\mathbf{V}}, \mathbf{W} \setminus \mathbf{H}\} \rangle$ iff it is identifiable wrt $\langle G, \mathbf{V}', \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$. If we consider each functional variable $W \in \mathbf{W}$, it is either in $\mathbf{V}'$ or having some parent that is not in $\mathbf{V}'$ (otherwise, W would have been added to $\mathbf{V}'$). By Lemma 4, the causal effect is identifiable wrt $\langle G, \mathbf{V}', \{\mathcal{C}_{\mathbf{V}}, \mathbf{W}\} \rangle$ iff it is identifiable wrt $\langle G, \mathbf{V}', \mathcal{C}_{\mathbf{V}} \rangle$. $\square$

APPENDIX C

Appendix for Chapter 4

C.1 Properties of Context-Induced Graphs

Context-induced graphs (CI-graphs) introduced in Definition 9 satisfy two key properties which we formalize in the following proposition.

Proposition 14. *Consider a causal graph G and a set of CSIs $\mathcal{C}$. Let T be a CI-graph under context $\mathbf{s}$. Then the following properties hold for all models Θ of G satisfying $\mathcal{C}$:*

- (i) *Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{S}$ be disjoint sets of variables. If $\mathbf{X}$ and $\mathbf{Y}$ are d-separated by $(\mathbf{Z} \cup \mathbf{S})$ in T , then $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}, \mathbf{s})$ in the distribution induced by Θ .¹*
- (ii) *For each variable V with parents $\mathbf{P}$ in G and parents $\mathbf{P}'$ in T , $\theta_{v|\mathbf{p}_1} = \theta_{v|\mathbf{p}_2}$ for all instantiations $\mathbf{p}_1, \mathbf{p}_2$ of $\mathbf{P}$ that extend a common $\mathbf{p}'$ that is compatible with $\mathbf{s}$.²*

The first property shows that CI-graphs preserve the conditional independencies given the context $\mathbf{s}$, and the second property implies that the parameters in CI-graphs characterize the equal parameters under the CSIs. In particular, we can always construct a model for the CI-graph by setting each parameter $\theta_{v|\mathbf{p}'}$ equal to $\theta_{v|\mathbf{p}}$ for any $\mathbf{p} \supseteq \mathbf{p}'$ (the value of $\theta_{v|\mathbf{p}'}$ is unique if $\mathbf{p}'$ is compatible with $\mathbf{s}$). The resulting model satisfies several important properties,

¹See [VP88] for details of d-separation and its interplay with conditional independencies.

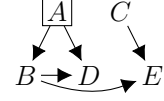
²Let $\mathbf{x}, \mathbf{y}$ be two instantiations. We say that $\mathbf{x}$ *extends* $\mathbf{y}$ if $\mathbf{x}$ contains all the states in $\mathbf{y}$, and that $\mathbf{x}$ and $\mathbf{y}$ are *compatible* if they do not contain conflicting states. The notion of CI-graph differs from the *context-specific graph* in [PNK15, THK19]: the latter satisfies property (i) but not property (ii).

including the preservation of $\Pr(\mathbf{v})$ for all $\mathbf{v}$ compatible with the context $\mathbf{s}$. We will use this construction technique to prove the soundness of our identification method (Algorithm 1).

We next illustrate the properties in Proposition 14 with an example. The figure on the right depicts a CI-graph of Figure 4.3(a) under context b_0 , assuming

CSIs $(D \perp\!\!\!\perp B \mid c_0, A)$, $(D \perp\!\!\!\perp C \mid b_0, A)$, and $(E \perp\!\!\!\perp B \mid c_0)$. Since A and E are d-separated by B in the CI-graph, $(A \perp\!\!\!\perp E \mid b_0)$ by property (i).

Moreover, $\theta_{D|A,b_0,c_1} = \theta_{D|A,b_0,c_2}$ holds for any states c_1, c_2 of C by property (ii).



C.2 Heuristics

We present heuristics that select a single choice for the CI-graph under $\mathbf{xy}$ (Algorithm 1, line 2), the context $\mathbf{s}$ and its CI-graph (Algorithm 1, lines 6-7), and the context-partition (Algorithm 2 line 2) when state constraints are imposed. Although the use of heuristics may compromise the completeness of the algorithms, it makes them computationally feasible by avoiding exponential enumerations and still substantially improves the identifiability of causal effects, as demonstrated by the empirical results in Section 4.5.

CI-graph under $\mathbf{xy}$ (Algorithm 1, line 2). We begin with a reverse-topological ordering Π over states $\mathbf{xy}$. Let $\mathbf{s}$ be a context initialized to be empty, and T be a graph initialized as the original graph G . We iterate over each state in Π in order, adding it to $\mathbf{s}$ and removing from T all edges enabled by the CSIs under the updated $\mathbf{s}$. The resulting graph T is guaranteed to be a CI-graph for $\mathbf{xy}$.

Context $\mathbf{s}$ and its CI-graph H (Algorithm 1, lines 6-7). We initialize the context $\mathbf{s}$ as follows. For each variable W that is in $\mathbf{C}_i$ or is a parent of some variable in $\mathbf{C}_i$ and has a state $w \in \mathbf{xy}$, we add w to $\mathbf{s}$. We initialize H as the original graph G . For each variable $C \in \mathbf{C}_i$ with some parent P , we remove the edge $P \rightarrow C$ from H if the edge is not in T .

The initialized $\mathbf{s}$ and H satisfy the condition in line 7.

We now augment the context $\mathbf{s}$ and prune additional edges from H in an iterative way. At each iteration, let $\mathbf{C} \supseteq \mathbf{C}_i$ be the c-component of H resulting from ID-QFUNC. If $\mathbf{C} = \mathbf{C}_i$, we terminate the iterative process since the c-factor of $\mathbf{C}_i$ is identifiable under $\mathbf{s}$ and H . Otherwise, we pick a CSI $(W \perp\!\!\!\perp P_1 \mid \mathbf{p}_2, \mathbf{P}_3)$ that satisfies the following conditions: $W \in (\mathbf{C} \setminus \mathbf{C}_i)$, $P_1 \in \mathbf{C}$, and $\mathbf{p}_2$ is compatible with $\mathbf{s}$. When multiple such CSIs exist, we prioritize the one whose P_1 is hidden and has fewest children in $H_{\mathbf{C}}$, where $H_{\mathbf{C}}$ denotes the subgraph of H only consisting of $\mathbf{C}$. We then add states $\mathbf{p}_2$ to the context $\mathbf{s}$ and remove the edge $P_1 \rightarrow C$ from H , before proceeding to the next iteration. The process terminates if no CSI satisfies the above conditions.

Context-partition (Algorithm 2, line 2). We construct the contexts in the partition and their corresponding CI-graphs simultaneously. We begin with a reverse-topological ordering Π over observed variables $\mathbf{V}$. Let $\mathbf{s}$ be a context initialized to be empty, and T be a graph initialized as the original graph G .

We iterate over each variable V in Π in order. At each iteration, let M be the mutilated graph obtained by removing $\mathbf{X}$ and retaining the ancestors of $(\mathbf{Y} \cup \mathbf{S})$ in T . If $V \in \mathbf{X} \cup \mathbf{Y}$ with state $v \in \mathbf{xy}$, we remove from T all edges enabled by the CSIs under the context $\mathbf{s} \cup \{v\}$, and continue the iteration using the current context $\mathbf{s}$ and the pruned graph.

If $V \notin \mathbf{X} \cup \mathbf{Y}$ but every state of V appears in the context of some CSI in M , we create a new context $\mathbf{s} \cup \{v_i\}$ for each state v_i of $\mathbf{V}$. For each new context, we assign a graph obtained by removing from T all edges enable by the CSIs under that new context. We then continue the iteration for each new context and its assigned graph.

Otherwise, we skip V and proceed with the current context $\mathbf{s}$ and graph T . After all variables have been processed, we collect the resulting contexts to form the context-partition and take their associated graphs as the CI-graphs.

C.3 Proofs

The proofs are presented in a different order from that of the main dissertation. We begin with the soundness proofs of the algorithms introduced in Section 4.4 and then establish the separation results in Sections 4.2 and 4.3.

Soundness Proofs

Proof of Proposition 14. We first show that property (ii) holds after each edge removal in the definition of CI-graph (Definition 9). Initially, $\mathbf{P}' = \mathbf{P}$ and hence $\theta_{v|\mathbf{p}_1} = \theta_{v|\mathbf{p}_2}$ for $\mathbf{p}_1 = \mathbf{p}_2$. Suppose now the property holds for graph T before the removal of edge $P_1 \rightarrow C$ based on the CSI ($C \perp\!\!\!\perp P_1 \mid \mathbf{p}_2, \mathbf{P}_3$). We want to show that property (ii) holds after the edge removal. Let $\mathbf{P}$ be the parents of C in T and $\mathbf{P}'$ be the parents after removal. Let $\mathbf{p}^1, \mathbf{p}^2$ be any two instantiations extending a common $\mathbf{p}'$ compatible with $\mathbf{s}$.

Suppose C appears in the context, i.e., $c^* \in \mathbf{s}$ for some state c^* . Then $\theta_{v|\mathbf{p}^1} = \theta_{v|\mathbf{p}_{c^*}^1}$ and $\theta_{v|\mathbf{p}^2} = \theta_{v|\mathbf{p}_{c^*}^2}$ according to the CSI constraint, where $\mathbf{p}_{c^*}^1, \mathbf{p}_{c^*}^2$ are obtained from $\mathbf{p}_1, \mathbf{p}_2$ by replacing the state of C by c^* . It follows that $\mathbf{p}_{c^*}^1, \mathbf{p}_{c^*}^2$ extend a common $\mathbf{p}$ that is compatible with $\mathbf{s}$. Hence, $\theta_{v|\mathbf{p}^1} = \theta_{v|\mathbf{p}_{c^*}^1} = \theta_{v|\mathbf{p}_{c^*}^2} = \theta_{v|\mathbf{p}^2}$ by induction.

Suppose C does not appear in the context, i.e., $C \notin \mathbf{S}$. Let c^* be an arbitrary state of C . Then $\theta_{v|\mathbf{p}^1} = \theta_{v|\mathbf{p}_{c^*}^1}$ and $\theta_{v|\mathbf{p}^2} = \theta_{v|\mathbf{p}_{c^*}^2}$ according to the CSI constraint. Since both $\mathbf{p}_{c^*}^1$ and $\mathbf{p}_{c^*}^2$ are compatible with $\mathbf{s}$, $\theta_{v|\mathbf{p}^1} = \theta_{v|\mathbf{p}_{c^*}^1} = \theta_{v|\mathbf{p}_{c^*}^2} = \theta_{v|\mathbf{p}^2}$ by induction.

We next prove property (i). For any model Θ for G that induces $\Pr$, we want to construct a model for T that induces a distribution $\Pr'$ satisfying $\Pr(\mathbf{v}) = \Pr'(\mathbf{v})$ for all instantiations $\mathbf{v} \supseteq \mathbf{s}$. This ensures that if d-separation holds, i.e., $\Pr'(\mathbf{X}|\mathbf{Y}, \mathbf{Z}, \mathbf{s}) = \Pr'(\mathbf{X}|\mathbf{Z}, \mathbf{s})$, the conditional independency also holds, i.e., $\Pr(\mathbf{X}|\mathbf{Y}, \mathbf{Z}, \mathbf{s}) = \Pr(\mathbf{X}|\mathbf{Z}, \mathbf{s})$. The construction can be done as follows. For each variable C with parents $\mathbf{P}'$ in T , we define its CPT entries as $\theta'_{c|\mathbf{p}'} = \theta_{c|\mathbf{p}}$ for every instantiation $(c, \mathbf{p}')$, where $\mathbf{p}$ can be any instantiation where $\mathbf{p} \supseteq \mathbf{p}'$. By property (ii), when $\mathbf{p}'$ is compatible with $\mathbf{s}$, the value of $\theta'_{c|\mathbf{p}'}$ is independent of the choice of $\mathbf{p}$ since all the

$\theta_{c|\mathbf{p}}$ are equal. Hence, $\Pr(\mathbf{v}) = \Pr'(\mathbf{v})$ for all $\mathbf{v}$ compatible with $\mathbf{s}$. $\square$

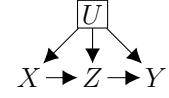
Before showing the soundness proofs for the algorithms, we first clarify the models (parameterizations) for CI-graphs so that the distributions induced by CI-graphs are well-defined. Let T be a CI-graph of the original graph G under some context $\mathbf{s}$ by Definition 9. We assume that the model Θ' of T is constructed from the model Θ of G as follows. For each variable C with parents $\mathbf{P}$ in G and parents $\mathbf{P}'$ in T , we assign $\theta'_{c|\mathbf{p}'} = \theta_{c|\mathbf{p}}$ for every instantiation $(c, \mathbf{p}')$, where $\mathbf{p}$ can be any instantiation satisfying $\mathbf{p} \supseteq \mathbf{p}'$; this is the same as the construction method shown in Appendix C.1. Based on the construction, we can now establish some key results on the c-factors induced by the CI-graph.

Lemma 5. *Consider a CI-graph T under the context $\mathbf{xy}$. Let M be the mutilated graph constructed in Algorithm 1 line 3 and let $\{\mathbf{C}_1, \dots, \mathbf{C}_m\}$ denote its maximal c-components. The causal effect $\Pr_{\mathbf{x}}(\mathbf{y})$ is identifiable if the c-factors $Q[\mathbf{C}_i]_{\mathbf{xy}}$ in T are identifiable for all $i = 1, \dots, m$.*

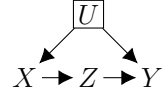
Proof. Let Θ be the model for the original graph G and Θ' be the model for the CI-graph T constructed using the above procedure. Let $\Pr$ and $\Pr'$ be the distributions induced by Θ and Θ' , respectively. By Proposition 14, Θ' agrees with Θ on all CPT entries that are compatible with $\mathbf{xy}$. As a result, $\Pr'_{\mathbf{x}}(\mathbf{v}) = \Pr_{\mathbf{x}}(\mathbf{v})$ for all instantiations $\mathbf{v} \supseteq \mathbf{y}$. Therefore, the CI-graph T can be used for causal identification since it preserves the value of causal effect, i.e., $\Pr'_{\mathbf{x}}(\mathbf{y}) = \Pr_{\mathbf{x}}(\mathbf{y})$. The techniques of graph mutilation and c-component (c-factor) decomposition follow from classical identification methods such as IDENTIFY [TP03]. $\square$

Proposition 15. *There exists a graph G and its CI-graph T under context $\mathbf{xy}$ for which $\Pr_{\mathbf{x}}(\mathbf{y})$ is unidentifiable wrt $\langle G, \mathbf{V}, \mathcal{C} \rangle$ but identifiable wrt $\langle T, \mathbf{V} \rangle$.*

Proof. Consider the causal graph G in Figure C.1(a) with the CSI constraint $(Z \perp\!\!\!\perp U \mid x)$. To compute the causal effect $\Pr_x(y)$, we construct the CI-graph T under xy as shown in Figure C.1(b). Applying classical identification methods (e.g., the front-door criterion [Pea95])



(a) Original G



(b) CI-graph T

Figure C.1: CI-graph constructed for the causal effect $\Pr_x(y)$ and CSI constraint $(Z \perp\!\!\!\perp U \mid x)$.

to T would suggest that the causal effect is identifiable. However, this conclusion is incorrect: we can construct the following parameterizations Θ^1, Θ^2 for G that satisfy the CSI constraint and induce the same $\Pr(\mathbf{V})$ yet yield distinct $\Pr_x(y)$.

U	θ_U^1	θ_U^2
u	0.5	0.4
$\bar{u}$	0.5	0.6

U	X	$\theta_{X U}^1$	$\theta_{X U}^2$
u	x	0.6	0.25
u	$\bar{x}$	0.4	0.75
$\bar{u}$	x	0.2	0.5
$\bar{u}$	$\bar{x}$	0.8	0.5

U	X	Z	$\theta_{Z U,X}^1$	$\theta_{Z U,X}^2$
u	x	z	0.5	0.5
u	x	$\bar{z}$	0.5	0.5
u	$\bar{x}$	z	0.3	1/3
u	$\bar{x}$	$\bar{z}$	0.7	2/3
$\bar{u}$	x	z	0.5	0.5
$\bar{u}$	x	$\bar{z}$	0.5	0.5
$\bar{u}$	$\bar{x}$	z	0.6	2/3
$\bar{u}$	$\bar{x}$	$\bar{z}$	0.4	1/3

U	Z	Y	$\theta_{Y U,Z}^1$	$\theta_{Y U,Z}^2$
u	z	y	0.5	0.76
u	z	$\bar{y}$	0.5	0.24
u	$\bar{z}$	y	0.8	0.42
u	$\bar{z}$	$\bar{y}$	0.2	0.58
$\bar{u}$	z	y	0.55	0.43
$\bar{u}$	z	$\bar{y}$	0.45	0.57
$\bar{u}$	$\bar{z}$	y	0.3	0.76
$\bar{u}$	$\bar{z}$	$\bar{y}$	0.7	0.24

□

Lemma 6. *Let T be a CI-graph of the original graph G under some context $\mathbf{s}$. For each c -component $\mathbf{C}$ in T (not necessarily maximal), its corresponding c -factor $Q[\mathbf{C}]_{\mathbf{s}}$ is identifiable if $\mathbf{C}$ can be obtained from T by repeatedly applying the following operations:³*

³For simplicity, we use T to refer to the graph and its set of variables interchangeably.

(i). Remove a leaf node that is not in $\mathbf{S}$; or

(ii). Replace T with the subgraph formed by one of its maximal c -components.

Proof. Consider again the construction method above Lemma 5. Let Θ and Θ' be the parameterizations for the original graph G and CI-graph T . Let $\Pr$ and $\Pr'$ be the distributions induced by Θ and Θ' . By Proposition 14 property (ii), the equality $\Pr(\mathbf{u}, \mathbf{v}) = \Pr'(\mathbf{u}, \mathbf{v})$ holds for all instantiations $(\mathbf{v}, \mathbf{u})$ compatible with $\mathbf{s}$.

We first extend the definition of c -factors to any variable sets $\mathbf{W}$ (not necessarily c -components): $Q[\mathbf{W}] = \sum_{\mathbf{U}_{\mathbf{W}}} \prod_{W \in \mathbf{W}} \theta_{W|\mathbf{P}_W}$, where $\mathbf{U}_{\mathbf{W}}$ are the hidden variables in $\mathbf{W}$ and $\mathbf{P}_W$ contains the parents of W in T . When $\mathbf{W}$ is a c -component in T , the definition is identical to the one in Appendix A.2. Moreover, we use $Q[\mathbf{W}]_{\mathbf{s}}$ to denote the instantiations of $Q[\mathbf{W}]$ compatible with the context $\mathbf{s}$.

Our goal is to show that the c -factors obtained from the two operations in this lemma are identifiable (computable from $\Pr(\mathbf{V})$). Let $\mathbf{W}$ be the result of applying the two operations. We prove the following statements by induction: (i) $Q[\mathbf{W}]_{\mathbf{s}}$ is identifiable; and (ii) $Q[\mathbf{W}^{(i)}]_{\mathbf{s}}$ is identifiable for each observed $W^i \in \mathbf{W}$, where W^i is the i^{th} variable in $\mathbf{W}$ based on some fixed topological order Π of G , $\mathbf{W}^{(i)}$ denotes all variables ordered before and including W^i , and $Q[\mathbf{W}^{(i)}]$ is defined as $\sum_{\mathbf{W} \setminus \mathbf{W}^{(i)}} Q[\mathbf{W}]$ [TP02b].

We start with the base case where $\mathbf{W}$ contains all variables in T , i.e., $\mathbf{W} = \mathbf{V} \cup \mathbf{U}$. By construction, $Q[\mathbf{W}]_{\mathbf{s}} = \Pr'(\mathbf{V})_{\mathbf{s}} = \Pr(\mathbf{V})_{\mathbf{s}}$ and hence statement (i) holds. Now consider statement (ii). For each observed $W^i \in \mathbf{W}$, all variables in $(\mathbf{W} \setminus \mathbf{W}^{(i)})$ can be pruned from both G and T . The CPT entries for the remaining variables $\mathbf{W}^{(i)}$ agree on all instantiations compatible with $\mathbf{s}$. Hence, $Q[\mathbf{W}^{(i)}]_{\mathbf{s}} = \Pr'(\mathbf{W}^{(i)} \cap \mathbf{V})_{\mathbf{s}} = \Pr(\mathbf{W}^{(i)} \cap \mathbf{V})_{\mathbf{s}}$ which is identifiable.

Suppose both statements hold for some variable set $\mathbf{W}$. Suppose a leaf $L \notin \mathbf{S}$ is pruned

from $\mathbf{W}$. Let $\mathbf{D}$ be the remaining variables, then the resulting c-factor $Q[\mathbf{D}]_{\mathbf{s}}$ is

$$Q[\mathbf{D}]_{\mathbf{s}} = \left(\sum_L Q[\mathbf{W}] \right)_{\mathbf{s}} = \sum_L Q[\mathbf{W}]_{\mathbf{s}}$$

By induction, the identifiability of $Q[\mathbf{D}]_{\mathbf{s}}$ follows from the identifiability $Q[\mathbf{W}]_{\mathbf{s}}$.

We next show that $Q[\mathbf{D}^{(i)}]$ is identifiable for each observed $D^i \in \mathbf{D}$ ordered based on Π . WLG, suppose D^i is ranked as the j^{th} variable in $\mathbf{W}$, i.e., $D^i = W^j$. We consider two cases based on the location of the pruned leaf L . If $L \in \mathbf{W}^{(j)}$, then

$$Q[\mathbf{D}^{(i)}]_{\mathbf{s}} = \left(\sum_L \sum_{\mathbf{W} \setminus \mathbf{W}^{(j)}} Q[\mathbf{W}] \right)_{\mathbf{s}} = \sum_L Q[\mathbf{W}^{(j)}]_{\mathbf{s}}$$

which can be computed by summing out L from $Q[\mathbf{W}^{(j)}]_{\mathbf{s}}$.

If $L \notin \mathbf{W}^{(j)}$, then

$$Q[\mathbf{D}^{(i)}]_{\mathbf{s}} = \left(\sum_{\mathbf{W} \setminus \mathbf{W}^{(j)}} Q[\mathbf{W}] \right)_{\mathbf{s}} = Q[\mathbf{W}^{(j)}]_{\mathbf{s}}$$

which is also equivalent to summing out L from $Q[\mathbf{W}^{(j)}]_{\mathbf{s}}$. Since $Q[\mathbf{W}^{(j)}]_{\mathbf{s}}$ is identifiable by induction, $Q[\mathbf{D}^{(i)}]_{\mathbf{s}}$ is also identifiable in both cases.

Suppose now $\mathbf{W}$ is replaced by a maximal c-component $\mathbf{D} \subset \mathbf{W}$. According to [TP03, Lemma 4], the c-factor for $\mathbf{D}$ can be computed as $Q[\mathbf{D}] = \prod_{W^i \in \mathbf{D} \cap \mathbf{V}} \frac{Q[\mathbf{W}^{(i)}]}{Q[\mathbf{W}^{(i-1)}]}$. Since each $Q[\mathbf{W}^{(i)}]_{\mathbf{s}}$ is identifiable by induction, $Q[\mathbf{D}]_{\mathbf{s}}$ is also identifiable. We next show that $Q[\mathbf{D}^{(i)}]_{\mathbf{s}}$ is identifiable for each observed $D^i \in \mathbf{D}$. By definition, $Q[\mathbf{D}^{(i)}] = \sum_{\mathbf{D} \setminus \mathbf{D}^{(i)}} Q[\mathbf{D}]$. To simplify the expression further, we iteratively eliminate observed variables in $(\mathbf{D} \setminus \mathbf{D}^{(i)})$ in reverse of Π (we do not need to eliminate hidden variables since they have already been summed out

in $Q[\mathbf{D}]$). WLG, let W^j be the first observed variable in the reverse order, then

$$\begin{aligned}
\sum_{W^j} Q[\mathbf{D}] &= \sum_{W^j} \prod_{W^k \in \mathbf{D} \cap \mathbf{V}} \frac{Q[\mathbf{W}^{(k)}]}{Q[\mathbf{W}^{(k-1)}]} \\
&= \left(\prod_{W^k \in \mathbf{D} \cap \mathbf{V}, k < j} \frac{Q[\mathbf{W}^{(k)}]}{Q[\mathbf{W}^{(k-1)}]} \right) \left(\sum_{W^j} \frac{Q[\mathbf{W}^{(j)}]}{Q[\mathbf{W}^{(j-1)}]} \right) \\
&= \prod_{W^k \in \mathbf{D} \cap \mathbf{V}, k < j} \frac{Q[\mathbf{W}^{(k)}]}{Q[\mathbf{W}^{(k-1)}]}
\end{aligned}$$

The second equality holds since W^j appears only in $\mathbf{W}^{(j)}$. We can apply the same procedure to eliminate all variables in $(\mathbf{D} \setminus \mathbf{D}^{(i)})$, yielding $Q[\mathbf{D}^{(i)}] = \prod_{W^k \in \mathbf{D}^{(i)} \cap \mathbf{V}} \frac{Q[\mathbf{W}^{(k)}]}{Q[\mathbf{W}^{(k-1)}]}$. Since each $Q[\mathbf{W}^{(k)}]_{\mathbf{s}}$ is identifiable by induction, $\prod_{W^k \in \mathbf{D}^{(i)} \cap \mathbf{V}} \frac{Q[\mathbf{W}^{(k)}]_{\mathbf{s}}}{Q[\mathbf{W}^{(k-1)}]_{\mathbf{s}}}$ is also identifiable. $\square$

Proof of Theorem 6. Suppose Algorithm 1 returns an identifying formula, there exists a CI-graph T under context $\mathbf{xy}$ such that the c-factors in the mutilated graph are deemed identifiable (lines 6-9). By Lemma 5, it suffices to show that a c-factor is indeed identifiable if ID-QFUNC returns an identifying formula for some context $\mathbf{s}$ and H in line 6.

Since $\mathbf{s}$ and H satisfy the condition in line 7, the c-factor $Q[\mathbf{C}_i]_{\mathbf{xy}}$ in T can be attained from the c-factor $Q[\mathbf{C}_i]_{\mathbf{s}}$ in H by selecting instantiations compatible with $\mathbf{xy}$. This is because the two c-factors involve the same set of CPTs, and the context $\mathbf{s}$ does not exclude any instantiations beyond those ruled out by the context $\mathbf{xy}$. Hence, $Q[\mathbf{C}_i]_{\mathbf{xy}}$ is identifiable if $Q[\mathbf{C}_i]_{\mathbf{s}}$ is identifiable. Since the identification method ID-QFUNC follows the two operations described in Lemma 6, the c-factor $Q[\mathbf{C}_i]_{\mathbf{s}}$ is identifiable if ID-QFUNC returns a formula for its estimation by the lemma. $\square$

Proof of Proposition 5. When no CSI is imposed, i.e., $\mathcal{C} = \emptyset$, the CI-graph is identical to the original causal graph G . By setting $\mathbf{s} = \emptyset$ and $H = G$ in line 6, Algorithm 1 becomes identical to the IDENTIFY algorithm in [TP03]. The operation of pruning leaves from $H_{\mathbf{D}}$ in lines 16-19 subsumes the operation of keeping the ancestors of $\mathbf{C}$ in IDENTIFY, and the operation of finding a maximal c-component in lines 20-22 is identical to the one in IDENTIFY.

Since IDENTIFY has been shown to be sound and complete [HV06], Algorithm 1 inherits the same properties for classical identifiability. $\square$

Proof of Theorem 7. Suppose there is a context-partition $\{\mathbf{s}_1, \dots, \mathbf{s}_k\}$ over $\mathbf{V} \setminus (\mathbf{X} \cup \mathbf{Y})$ such that each $\Pr_{\mathbf{x}}(\mathbf{y}, \mathbf{s}_i)$ is identifiable by Algorithm 1. Then $\Pr_{\mathbf{x}}(\mathbf{y}) = \sum_{i=1}^k \Pr_{\mathbf{x}}(\mathbf{y}, \mathbf{s}_i)$ is identifiable by the law of total probability. $\square$

Separation Results

Proof of Proposition 3. The only-if direction directly follows from the definitions. We now show the contrapositive of the if direction. Suppose $\Pr_{\mathbf{X}}(\mathbf{Y})$ is unidentifiable, there exist some $\mathbf{x}'$ and $\mathbf{y}'$ such that $\Pr_{\mathbf{x}'}(\mathbf{y}')$ is unidentifiable. We can simply set $\mathbf{x} = \mathbf{x}'$ and $\mathbf{y} = \mathbf{y}'$ for the states in $\Pr_{\mathbf{x}}(\mathbf{y})$. $\square$

Proof of Proposition 4. If $\Pr_{\mathbf{x}}(\mathbf{y})$ is unidentifiable, then $\Pr_{\mathbf{X}}(\mathbf{Y})$ is also unidentifiable by the definition of variable-based identifiability. Now consider the opposite direction. If $\Pr_{\mathbf{X}}(\mathbf{Y})$ is unidentifiable, then there exist states $\mathbf{x}$ and $\mathbf{y}$ such that $\Pr_{\mathbf{x}}(\mathbf{y})$ is unidentifiable. We go over each state s in $\mathbf{x}\mathbf{y}$ and create a new state s' (outside CSI contexts) for S . The CPTs for S and each of its children C are updated as follows:

$$\theta'_{s'|\mathbf{p}} = \theta_{s|\mathbf{p}} \cdot \epsilon, \quad \theta'_{s|\mathbf{p}} = \theta_{s|\mathbf{p}} \cdot (1 - \epsilon), \quad \theta'_{c|\mathbf{p}_C, s'} = \theta_{c|\mathbf{p}_C, s}$$

where $\mathbf{P}_C$ denotes the parents of C (excluding S) and ϵ is a positive constant in range $(0, 1)$. The construction ensures that all CSIs satisfied by the original parameterization Θ are also satisfied by the new parameterization Θ' .

Let $\mathbf{x}'$ and $\mathbf{y}'$ be the results of replacing s by s' in $\mathbf{x}$ and $\mathbf{y}$. Then the update parameter-

ization induces the following observational distribution $\Pr'(\mathbf{V})$ and causal effect $\Pr'_{\mathbf{x}'}(\mathbf{y}')$:

$$\Pr'(\mathbf{v}) = \begin{cases} \Pr(\mathbf{v}|_s) \cdot \epsilon & s' \in \mathbf{v} \\ \Pr(\mathbf{v}) \cdot (1 - \epsilon) & s \in \mathbf{v} \\ \Pr(\mathbf{v}) & \text{otherwise} \end{cases}$$

$$\Pr'_{\mathbf{x}'}(\mathbf{y}') = \begin{cases} \Pr_{\mathbf{x}}(\mathbf{y}) \cdot \epsilon & S \in \mathbf{Y} \\ \Pr_{\mathbf{x}}(\mathbf{y}) & \text{otherwise} \end{cases}$$

where $\mathbf{v}|_s$ replaces the state of S in $\mathbf{v}$ by the state s . Hence, if there exist parameterizations Θ^1, Θ^2 that induce a same $\Pr(\mathbf{V})$ yet distinct $\Pr_{\mathbf{x}}(\mathbf{y})$, we can apply the above procedure to construct parameterizations that induce a same $\Pr'(\mathbf{V})$ yet distinct $\Pr_{\mathbf{x}'}(\mathbf{y}')$. By iterating over all states in $\mathbf{x}\mathbf{y}$, we conclude that there exist $\mathbf{x}', \mathbf{y}'$ that do not involve any contexts of CSIs and for which $\Pr_{\mathbf{x}'}(\mathbf{y}')$ is unidentifiable. We finally set $\mathbf{x} = \mathbf{x}'$ and $\mathbf{y} = \mathbf{y}'$ for $\Pr_{\mathbf{x}}(\mathbf{y})$. $\square$

Proposition 16. *There exists a tuple $\langle G, \mathbf{V}, \mathcal{C} \rangle$ where $\mathcal{C}$ contains only CSI constraints for which $\Pr_{\mathbf{x}}(\mathbf{Y})$ is unidentifiable but $\Pr_{\mathbf{x}}(\mathbf{y})$ is identifiable for some states $\mathbf{x}, \mathbf{y}$.*

Proof. Consider the causal graph in Figure 4.1(a). Let “entry-level” be state 0 of J , “ $Y \geq 10$ ” be state 0 of Y , and “low” be state 0 of S . The CSI constraint can be written as $(S \perp\!\!\!\perp D | J = 0, Y)$.

We apply Algorithm 1 to show the identifiability of $\Pr_{Y=0}(J = 0, S = 0)$. Figure 4.4(a) depicts the CI-graph T under the context $\{Y = 0, J = 0, S = 0\}$ and Figure 4.4(b) depicts its mutilated graph. We are left to show that the c-factors $Q[ADJ]_{Y=0, J=0, S=0}$ and $Q[S]_{Y=0, J=0, S=0}$ in the mutilated graph are identifiable.

To identify the first c-factor, we consider the context $\mathbf{s} : \{J = 0\}$ and its corresponding CI-graph which coincides with T . Since the context and CI-graph satisfy the condition in line 7 of Algorithm 1, we can apply the ID-QFUNC procedure to establish the identifiability

of $Q[ADJ]_{J=0}$. This is because the c-component $\{A, D, J\}$ can be obtained from CI-graph T by first finding the maximal c-component $\{A, D, J, Y\}$ and then pruning the leaf node Y . To identify the second c-factor, we again consider the context $\mathbf{s} : \{J = 0\}$ and its CI-graph depicted in T . Since $\{S\}$ is directly a maximal c-component in T , the c-factor $Q[S]_{J=0}$ is identifiable according to ID-QFUNC. Together, the causal effect $\Pr_{Y=0}(J = 0, S = 0)$ is identifiable since the entries in c-factors $Q[ADJ]_{Y=0, J=0, S=0}$ and $Q[S]_{Y=0, J=0, S=0}$ can be extracted from $Q[ADJ]_{J=0}$ and $Q[S]_{J=0}$.

We next show that the variable-based causal effect $\Pr_Y(J, S)$ is not identifiable. We assume that J has three states $\{0, 1, 2\}$ and all other variables are binary with states $\{0, 1\}$. For both parameterizations Θ_1, Θ_2 , we assign uniform distributions for A, D , structural equation $Y = A$, and the following CPT for J :

$$\theta_{J|A,D}^1 = \theta_{J|A,D}^2 = \begin{cases} 0.01 & \text{if } J = 0 \\ 0.99 & \text{if } A \oplus D = J - 1 \\ 0 & \text{if } A \oplus D \neq J - 1 \end{cases}$$

we then assign different CPTs for S :

$$\theta_{S|D,Y,J}^1 = \begin{cases} 0.5 & \text{if } J = 0 \\ 0.99 & \text{if } J \neq 0 \text{ and } (J - 1) \oplus D \oplus Y = S \\ 0.01 & \text{if } J \neq 0 \text{ and } (J - 1) \oplus D \oplus Y \neq S \end{cases}$$

$$\theta_{S|D,Y,J}^2 = \begin{cases} 0.5 & \text{if } J = 0 \\ 0.99 & \text{if } J \neq 0 \text{ and } S = 0 \\ 0.01 & \text{if } J \neq 0 \text{ and } S = 1 \end{cases}$$

The two parameterizations induce a same $\Pr(\mathbf{V})$ yet different $\Pr_{Y=0}(J = 1, S = 1)$. $\square$

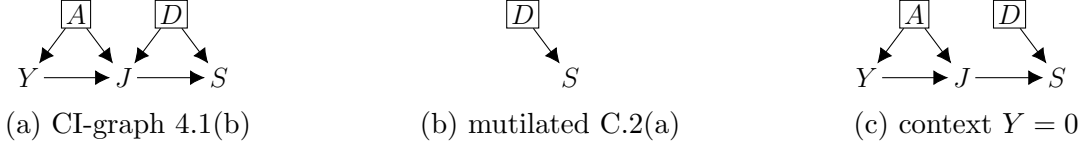


Figure C.2: Context-induced and mutilated graphs for the examples in Figure 4.1 under the CSI ($J \perp\!\!\!\perp D \mid A, Y = 0$).

Proposition 17. *There exists a tuple $\langle G, \mathbf{V}, \mathcal{C} \rangle$ where $\mathcal{C}$ contains only CSI constraints for which $\Pr_{\mathbf{X}}(\mathbf{Y})$ is unidentifiable wrt $\langle G, \mathbf{V} \rangle$ but is identifiable wrt $\langle G, \mathbf{V}, \mathcal{C} \rangle$.*

Proof. Consider the causal graph in Figure 4.1(b) and the CSI ($J \perp\!\!\!\perp D \mid A, Y = 0$). The classical causal effect $\Pr_J(S)$ is unidentifiable by the ID algorithm [SP06] since the subgraph containing $\{A, Y, J\}$ forms a *hedge* wrt the causal effect.⁴

We next show that the causal effect is identifiable with the CSI constraint. By Proposition 4, the variable-based causal effect $\Pr_J(S)$ is identifiable iff the state-based causal effect $\Pr_{J=0}(S = 0)$ is identifiable since none of the contexts appear in the the CSI. The CI-graph under the context $\{J = 0, S = 0\}$ is identical to the original causal graph, and its mutilated graph T is depicted in Figure C.2(b). The only c-component in the mutilated graph is $\{S, D\}$ and its corresponding c-factor is $Q[SD]_{J=0, S=0}$. We now construct the context $\mathbf{s} : \{Y = 0\}$ with the corresponding CI-graph H shown in Figure C.2(c). Since $\mathbf{s}$ and H satisfy the condition in line 7 of Algorithm 1, $Q[SD]_{J=0, S=0}$ is identifiable if $Q[SD]_{Y=0}$ is identifiable by the ID-QFUNC procedure. This is indeed the case since the c-component $\{S, D\}$ is a maximal c-component in H . $\square$

Before showing Proposition 18, we present the following Lemma 7 which will be used for the proof of Proposition 18.

Lemma 7. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two sets of state constraints where $\mathcal{C}_1 \subseteq \mathcal{C}_2$.⁵ Then a causal*

⁴A *hedge* is a graphical notion that characterizes the unidentifiability of causal effects; see [SP06] for details.

⁵We say $\mathcal{C}_1 \subseteq \mathcal{C}_2$ if for each variable T , the states of T in $\mathcal{C}_1$ are a subset of the states of T in $\mathcal{C}_2$.

effect $\text{Pr}_{\mathbf{x}}(\mathbf{y})$ is identifiable wrt $\langle G, \mathbf{V}, \mathcal{C}_1 \rangle$ if it is identifiable wrt $\langle G, \mathbf{V}, \mathcal{C}_2 \rangle$.

Proof. By induction, it suffices to show that an unidentifiable causal effect $\text{Pr}_{\mathbf{x}}(\mathbf{y})$ remains unidentifiable if we add a new state to any variable W . Since $\text{Pr}_{\mathbf{x}}(\mathbf{y})$ is unidentifiable, there exist two parameterizations Θ_1, Θ_2 that induce a same $\text{Pr}(\mathbf{V})$ but different values for $\text{Pr}_{\mathbf{x}}(\mathbf{y})$. Our proof is based on constructing parameterizations Θ'_1, Θ'_2 that contain an additional state for W while inducing unidentifiability.

Let $\mathbf{P} = \{P_1, \dots, P_k\}$ and $\mathbf{C} = \{C_1, \dots, C_t\}$ be the parents and children of W . Moreover, let $w_1, \dots, w_m$ be the current states and w' be the new state of W . We construct new parameterizations Θ'_1, Θ'_2 from the original parameterizations Θ_1, Θ_2 as follows. Let ϵ be a positive constant in $(0, 1)$. We modify the CPTs for W and each child $C \in \mathbf{C}$:

$$\theta'_{w'|\mathbf{p}} = \theta_{w_1|\mathbf{p}} \cdot \epsilon \quad \theta'_{w_1|\mathbf{p}} = \theta_{w_1|\mathbf{p}} \cdot (1 - \epsilon)$$

$$\theta'_{c|\mathbf{p}_C, w'} = \theta_{c|\mathbf{p}_C, w_1}$$

where $\mathbf{p}_C$ denotes the parents of C excluding W . Let Pr'_1 and Pr'_2 be the distribution induced by Θ'_1 and Θ'_2 . For each instantiation $(\mathbf{u}, \mathbf{v})$ over hidden and observed variables,

$$\text{Pr}'(\mathbf{u}, \mathbf{v}) = \begin{cases} \text{Pr}((\mathbf{u}, \mathbf{v})|_{w_1}) \cdot \epsilon & w' \in (\mathbf{u}, \mathbf{v}) \\ \text{Pr}(\mathbf{u}, \mathbf{v}) \cdot (1 - \epsilon) & w_1 \in (\mathbf{u}, \mathbf{v}) \\ \text{Pr}(\mathbf{u}, \mathbf{v}) & \text{otherwise} \end{cases}$$

where $(\mathbf{u}, \mathbf{v})|_{w_1}$ replaces the state of W in $(\mathbf{u}, \mathbf{v})$ by w_1 . Suppose W is hidden, summing out $\mathbf{U}$ yields $\text{Pr}'(\mathbf{V}) = \text{Pr}(\mathbf{V})$; hence, $\text{Pr}'_1(\mathbf{V}) = \text{Pr}_1(\mathbf{V}) = \text{Pr}_2(\mathbf{V}) = \text{Pr}'_2(\mathbf{V})$. Suppose W is

observed, we can compute $\Pr'(\mathbf{V})$ as follows:

$$\Pr'(\mathbf{v}) = \begin{cases} \Pr(\mathbf{v}|_{w_1}) \cdot \epsilon & w' \in \mathbf{v} \\ \Pr(\mathbf{v}) \cdot (1 - \epsilon) & w_1 \in \mathbf{v} \\ \Pr(\mathbf{v}) & \text{otherwise} \end{cases}$$

Again, $\Pr'_1(\mathbf{V}) = \Pr'_2(\mathbf{V})$ since $\Pr_1(\mathbf{V}) = \Pr_2(\mathbf{V})$.

We are left to show that $\Pr'_{1\mathbf{x}}(\mathbf{y}) \neq \Pr'_{2\mathbf{x}}(\mathbf{y})$. Suppose $W \in \mathbf{U}$, the distribution $\Pr_{\mathbf{x}}(\mathbf{V})$ is preserved and therefore $\Pr'_{1\mathbf{x}}(\mathbf{y}) = \Pr_{1\mathbf{x}}(\mathbf{y}) \neq \Pr_{2\mathbf{x}}(\mathbf{y}) = \Pr'_{2\mathbf{x}}(\mathbf{y})$. Suppose $W \in \mathbf{V}$, we can again compute $\Pr'_{\mathbf{x}}(\mathbf{y})$ as follows:

$$\Pr'_{\mathbf{x}}(\mathbf{y}) = \begin{cases} \Pr_{\mathbf{x}}(\mathbf{y}) & W \notin \mathbf{Y} \\ \Pr_{\mathbf{x}}(\mathbf{y}|_{w_1}) \cdot \epsilon & W \in \mathbf{Y} \text{ and } w' \in \mathbf{y} \\ \Pr_{\mathbf{x}}(\mathbf{y}) \cdot (1 - \epsilon) & W \in \mathbf{Y} \text{ and } w_1 \in \mathbf{y} \end{cases}$$

In all cases, $\Pr'_{1\mathbf{x}}(\mathbf{y}) \neq \Pr'_{2\mathbf{x}}(\mathbf{y})$. □

Proposition 18. *Let $\mathcal{C}$ be a set of state constraints and G be a Semi-Markovian graph. A causal effect $\Pr_{\mathbf{x}}(\mathbf{Y})$ (or $\Pr_{\mathbf{x}}(\mathbf{y})$) is identifiable wrt $\langle G, \mathbf{V} \rangle$ iff $\Pr_{\mathbf{x}}(\mathbf{Y})$ (or $\Pr_{\mathbf{x}}(\mathbf{y})$) is constrained-identifiable wrt $\langle G, \mathbf{V}, \mathcal{C} \rangle$.*

Proof. The only-if direction follows from the definition. We now prove the contrapositive of the if direction. According to the ID algorithm [SP06], $\Pr_{\mathbf{x}}(\mathbf{Y})$ is unidentifiable iff there exist two parameterizations Θ^1 and Θ^2 that induce a same $\Pr(\mathbf{V})$ yet different $\Pr_{\mathbf{x}}(\mathbf{y})$ and in which *all variables are binary*. Hence, we can apply Lemma 7 to construct parameterizations with arbitrary numbers of variable states that induce unidentifiability. The equivalence between $\Pr_{\mathbf{x}}(\mathbf{Y})$ and $\Pr_{\mathbf{x}}(\mathbf{y})$ follows from the fact that we can permute the states of $\mathbf{X}, \mathbf{Y}$. □

Proposition 19. *There exist a causal graph G , observed variables $\mathbf{V}$, CSI constraints $\mathcal{C}_1$,*

and state constraints $\mathcal{C}_2$ for which $\Pr_{\mathbf{X}}(\mathbf{Y})$ is unidentifiable wrt $\langle G, \mathbf{V}, \mathcal{C}_1 \rangle$ but is identifiable wrt $\langle G, \mathbf{V}, \mathcal{C}_1 \cup \mathcal{C}_2 \rangle$.

Proof. Consider the causal graph in Figure 4.2(a) and the CSI constraints $(X \perp\!\!\!\perp U|A = 0)$ and $(Y \perp\!\!\!\perp U|X, A = 1)$. We first show that the variable-based causal effect is unidentifiable without restricting variable states. Assume variables U, X, Y have states $\{0, 1\}$ and A has states $\{0, 1, 2\}$. We construct two parameterizations Θ^1, Θ^2 that induce a same $\Pr(\mathbf{V})$ yet distinct answers for the causal effect $\Pr_{X=0}(Y = 0)$. In both parameterizations, we assign uniform distributions for U, A and the following CPT for X :

$$\theta_{X|U,A}^1 = \theta_{X|U,A}^2 = \begin{cases} 0.5 & \text{if } A = 0 \text{ or } A = 1 \\ 1 & \text{if } A = 2 \text{ and } X = U \\ 0 & \text{if } A = 2 \text{ and } X \neq U \end{cases}$$

We then assign different CPTs for Y as follows:

$$\theta_{Y|U,X,A}^1 = \begin{cases} 0.5 & \text{if } A = 0 \text{ or } A = 1 \\ 0.99 & \text{if } A = 2 \text{ and } U \oplus X = Y \\ 0.01 & \text{if } A = 2 \text{ and } U \oplus X \neq Y \end{cases}$$

$$\theta_{Y|U,X,A}^2 = \begin{cases} 0.5 & \text{if } A = 0 \text{ or } A = 1 \\ 0.99 & \text{if } A = 2 \text{ and } Y = 0 \\ 0.01 & \text{if } A = 2 \text{ and } Y = 1 \end{cases}$$

The two parameterizations induce a same $\Pr(\mathbf{V})$ yet distinct $\Pr_{X=0}(Y)$ and $\Pr_{X=1}(Y)$.

We next apply Algorithm 2 to show that the causal effect $\Pr_X(Y)$ is identifiable if A has states $\{0, 1\}$. We consider the context-partition $\{A=0, A=1\}$ and reduce the problem of

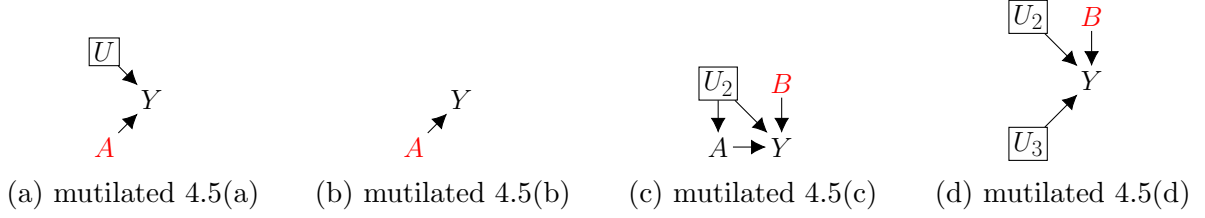


Figure C.3: Mutilated graphs for the examples in Figure 4.5.

identifying $\Pr_X(Y)$ to that of identifying sub-queries $\Pr_{X=0}(Y = 0, A = 0)$ and $\Pr_{X=0}(Y = 0, A = 1)$. Figures 4.5(a) and 4.5(b) depict the CI-graphs for the two sub-queries, and Figures C.3(a) and C.3(b) depict their mutilated graphs.

To identify the two c-factors in Figures C.3(a), we consider the context $A=0$ and its CI-graph T which coincides with Figure 4.5(a). Since $\{U, Y\}$ and $\{A\}$ are maximal c-components in T , their c-factors $Q[UY]_{A=0}$ and $Q[A]_{A=0}$ are identifiable. It follows that $Q[UY]_{A=0, X=0, Y=0}$ and $Q[A]_{A=0, X=0, Y=0}$ are also identifiable, which concludes the identifiability of $\Pr_{X=0}(Y = 0, A = 0)$.

To identify the two c-factors in Figures C.3(b), we consider the context $A=1$ and its CI-graph T which coincides with Figure 4.5(b). Since both $\{Y\}$ and $\{A\}$ are c-components in T , their corresponding c-factors $Q[Y]_{A=1}$ and $Q[A]_{A=1}$ are computable from $\Pr(\mathbf{V})$, which establishes the identifiability of $\Pr_{X=0}(Y = 0, A = 1)$. $\square$

Proposition 20. *There exist a causal graph G , observed variables $\mathbf{V}$, CSI constraints $\mathcal{C}_1$, state constraints $\mathcal{C}_2$, and particular states $\mathbf{x}$, and $\mathbf{y}$ where $\Pr_{\mathbf{x}}(\mathbf{y})$ is unidentifiable wrt $\langle G, \mathbf{V}, \mathcal{C}_1 \rangle$ but is identifiable wrt $\langle G, \mathbf{V}, \mathcal{C}_1 \cup \mathcal{C}_2 \rangle$. Moreover, $\Pr_{\mathbf{x}}(\mathbf{Y})$ is unidentifiable wrt $\langle G, \mathbf{V}, \mathcal{C}_1 \cup \mathcal{C}_2 \rangle$.*

Proof. Consider the second example for Figure 4.2(b) in Section 4.3. For simplicity, let *rainy* be the state 0 and *sunny* be the state 1 of B , and let *low-cost* be the state 0 of X . We can rewrite the CSIs as $(A \perp\!\!\!\perp U_1 \mid X = 0, U_2)$, $(Y \perp\!\!\!\perp U_3 \mid B = 0, U_2, A)$, and $(Y \perp\!\!\!\perp A \mid B = 1, U_2, U_3)$.

We first show $\Pr_{X=0}(Y = 0)$ is unidentifiable by finding two parameterizations Θ^1, Θ^2 that induce a same $\Pr(\mathbf{V})$ yet different $\Pr_{X=0}(Y = 0)$. We assume that all variables have states $\{0, 1\}$, except for B , which has states $\{0, 1, 2\}$. For both parameterizations, we assign uniform distributions for U_1, U_2, U_3 , structural equations $X = U_3$, $A = U_2 \oplus X$, and the following CPT for B :

$$\theta_B^1 = \theta_B^2 = \begin{cases} 0.05 & \text{if } B = 0 \text{ or } B = 1 \\ 0.9 & \text{if } B = 2 \end{cases}$$

We assign different CPTs for Y in Θ_1 and Θ_2 :

$$\theta_{Y|U_2, U_3, A, B}^1 = \begin{cases} 0.5 & \text{if } B = 0 \text{ or } B = 1 \\ 0.99 & \text{if } B = 1 \text{ and } Y = U_2 \oplus A \oplus U_3 \\ 0.01 & \text{if } B = 1 \text{ and } Y \neq U_2 \oplus A \oplus U_3 \end{cases}$$

$$\theta_{Y|U_2, U_3, A, B}^2 = \begin{cases} 0.5 & \text{if } B = 0 \text{ or } B = 1 \\ 0.99 & \text{if } B = 1 \text{ and } Y = 0 \\ 0.01 & \text{if } B = 1 \text{ and } Y = 1 \end{cases}$$

The two parameterizations attains a same $\Pr(\mathbf{V})$ yet different answers for $\Pr_{X=0}(Y = 0)$.

We now apply Algorithm 2 to show that $\Pr_{X=0}(Y = 0)$ is identifiable if the states of B are restricted to $\{0, 1\}$. We consider the context-partition $\{B=0, B=1\}$ and show that the sub-queries $\Pr_{X=0}(Y = 0, B = 0)$ and $\Pr_{X=0}(Y = 0, B = 1)$ are identifiable. Figures 4.5(c) and 4.5(d) depict the CI-graphs for the sub-queries, and Figures C.3(c) and C.3(d) depict their mutilated graphs.

We first identify the c-factors of c-components $\{U_2, A, Y\}$ and $\{B\}$ in the mutilated graph for sub-query $\Pr_{X=0}(Y = 0, B = 0)$. Consider the context $\mathbf{s} : \{X=0, B=0\}$ and its CI-graph H which coincides with Figure C.3(c) (Algorithm 1 lines 6-7). Since both $\{U_2, A, Y\}$ and $\{B\}$ are maximal c-components in H , $\Pr_{X=0}(Y = 0, B = 0)$ is identifiable.

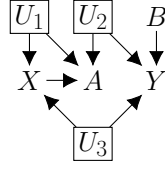


Figure C.4: CI-graph of Figure 4.2(b) under the context $B=1$.

We now identify the c-factors of c-components $\{U_2, U_3, Y\}$ and $\{B\}$ in the mutilate graph for sub-query $\Pr_{X=0}(Y = 0, B = 1)$. We consider the context $\mathbf{s} : \{B=1\}$ and its CI-graph H shown in Figure C.4. The first c-component can be obtained from H by first pruning leaves A, X and then taking the maximal c-component. The second c-component is a maximal c-component of H . Hence, $\Pr_{X=0}(Y = 0, B = 1)$ is identifiable by the soundness of Algorithm 1.

We finally show that $\Pr_X(Y)$ is unidentifiable under both the CSI and state constraints by constructing parameterizations Θ^1, Θ^2 that form an instance of unidentifiability. We assume all variables are binary with states $\{0, 1\}$, except for X , which has states $\{0, 1, 2\}$. For both Θ^1 and Θ^2 , we assign uniform distributions for variables U_1, U_2, U_3, B and the following CPTs for X, Y :

$$\theta_{X|U_1, U_3}^1 = \theta_{X|U_1, U_3}^2 = \begin{cases} (0.05, 0.95, 0) & \text{if } U_1 = 0 \\ (0.05, 0, 0.95) & \text{if } U_1 = 1 \end{cases}$$

$$\theta_{Y|U_2, U_3, A, B}^1 = \theta_{Y|U_2, U_3, A, B}^2 = \begin{cases} 0.99 & \text{if } A = Y \\ 0.01 & \text{if } A \neq Y \end{cases}$$

We assign distinct CPTs for A in Θ^1 and Θ^2 :

$$\theta_{A|U_1, U_2, X}^1 = \begin{cases} 0.5 & \text{if } X = 0 \\ 0.99 & \text{if } X \neq 0 \text{ and } (X - 1) \oplus U_1 = A \\ 0.01 & \text{if } X \neq 0 \text{ and } (X - 1) \oplus U_1 \neq A \end{cases}$$

$$\theta_{A|U_1, U_2, X}^2 = \begin{cases} 0.5 & \text{if } X = 0 \\ 0.99 & \text{if } X \neq 0 \text{ and } A = 0 \\ 0.01 & \text{if } X \neq 0 \text{ and } A = 1 \end{cases}$$

The parameterizations induce a same $\Pr(\mathbf{V})$ yet different values for $\Pr_{X=1}(Y = 1)$. $\square$

Proposition 21. *Let $\mathcal{C}_1$ be a set of CSI constraints and $\mathcal{C}_2$ be a set of state constraints. Suppose $\mathcal{C}_2$ mentions treatment variables $\mathbf{X}_1$ and outcome variables $\mathbf{Y}_1$. Let $\mathbf{X}_2 = \mathbf{X} \setminus \mathbf{X}_1$ and $\mathbf{Y}_2 = \mathbf{Y} \setminus \mathbf{Y}_1$. Then $\Pr_{\mathbf{X}}(\mathbf{Y})$ is identifiable (wrt $\langle G, \mathbf{V}, \mathcal{C}_1 \cup \mathcal{C}_2 \rangle$) iff $\Pr_{\mathbf{x}_1 \mathbf{x}_2}(\mathbf{y}_1 \mathbf{y}_2)$ is identifiable for all assignments $\mathbf{x}_1, \mathbf{y}_1$ specified by $\mathcal{C}_2$ and any assignment $\mathbf{x}_2, \mathbf{y}_2$ that do not appear in the contexts of $\mathcal{C}_1$.*

Proof. The only-if direction follows from the definition of variable-based identifiability (Definition 3). We now prove the contrapositive of the if direction. Suppose $\Pr_{\mathbf{X}}(\mathbf{Y})$ is unidentifiable, there exist two parameterizations that induce a same $\Pr(\mathbf{V})$ but different $\Pr_{\mathbf{x}_1 \mathbf{x}_2}(\mathbf{y}_1 \mathbf{y}_2)$, where $\mathbf{x}_1, \mathbf{y}_1$ are states specified by $\mathcal{C}_2$. We then iterate over each state s in $\mathbf{x}_2 \mathbf{y}_2$ and apply the construction method from the proof of Proposition 4. The resulting parameterizations satisfy $\mathcal{C}_1$ and $\mathcal{C}_2$, induce a same $\Pr(\mathbf{V})$, and yield different values of $\Pr_{\mathbf{x}_1 \mathbf{x}'_2}(\mathbf{y}_1 \mathbf{y}'_2)$ for $\mathbf{x}'_2, \mathbf{y}'_2$ outside the CSI contexts. $\square$

Corollary 8. *Let $\mathcal{C}_1$ be a set of CSI constraints and $\mathcal{C}_2$ be a set of state constraints. If the states $\mathbf{x}, \mathbf{y}$ do not appear in the context of $\mathcal{C}_1$ and variables $\mathbf{X}, \mathbf{Y}$ are not mentioned in $\mathcal{C}_2$, then $\Pr_{\mathbf{X}}(\mathbf{Y})$ is identifiable wrt $\langle G, \mathbf{V}, \mathcal{C}_1 \cup \mathcal{C}_2 \rangle$ iff $\Pr_{\mathbf{x}}(\mathbf{y})$ is identifiable wrt $\langle G, \mathbf{V}, \mathcal{C}_1 \cup \mathcal{C}_2 \rangle$.*

Proof. Since $\mathcal{C}_2$ does not mention variables $\mathbf{X}$ and $\mathbf{Y}$, the variable sets in Proposition 21 take the following values: $\mathbf{X}_1 = \emptyset$, $\mathbf{Y}_1 = \emptyset$, $\mathbf{X}_2 = \mathbf{X}$, and $\mathbf{Y}_2 = \mathbf{Y}$. Hence, $\Pr_{\mathbf{X}}(\mathbf{Y})$ is identifiable iff $\Pr_{\mathbf{x}}(\mathbf{y})$ is identifiable for any $\mathbf{x}, \mathbf{y}$ not appearing in the CSI contexts. $\square$

C.4 Complete Experimental Results

n	r	stats	CLASSICAL-ID	ID-CSI-HEU		ID-CSI-STATE-HEU		DOSEARCH
			variable & state	variable	state	variable	state	variable
6	0.2	#ID/success	171/200	171/200	171/200	171/200	171/200	171/200
		runtime (s)	0.001	0.001	0.001	0.002	0.002	0.9
	0.5	#ID/success	171/200	172/200	174/200	173/200	176/200	173/200
		runtime (s)	0.001	0.001	0.002	0.002	0.002	0.9
	0.8	#ID/success	171/200	171/200	172/200	175/200	176/200	174/200
		runtime (s)	0.001	0.001	0.001	0.002	0.002	0.9
8	0.2	#ID/success	120/200	120/200	123/200	121/200	124/200	120/200
		runtime (s)	0.002	0.002	0.003	0.003	0.002	6.2
	0.5	#ID/success	120/200	121/200	132/200	123/200	133/200	122/199
		runtime (s)	0.002	0.002	0.002	0.005	0.003	4.9
	0.8	#ID/success	120/200	121/200	133/200	126/200	137/200	123/198
		runtime (s)	0.002	0.002	0.002	0.005	0.003	6.6
10	0.2	#ID/success	115/200	115/200	118/200	115/200	118/200	112/198
		runtime (s)	0.003	0.004	0.003	0.004	0.003	14.0
	0.5	#ID/success	115/200	117/200	127/200	120/200	132/200	116/191
		runtime (s)	0.002	0.003	0.003	0.005	0.005	42.5
	0.8	#ID/success	115/200	119/200	132/200	129/200	140/200	124/190
		runtime (s)	0.002	0.003	0.003	0.007	0.004	40.2
12	0.2	#ID/success	107/200	108/200	113/200	109/200	114/200	99/175
		runtime (s)	0.003	0.003	0.004	0.005	0.004	92.2
	0.5	#ID/success	107/200	109/200	121/200	114/200	124/200	79/122
		runtime (s)	0.003	0.003	0.003	0.007	0.005	165.3
	0.8	#ID/success	107/200	109/200	131/200	121/200	136/200	79/127
		runtime (s)	0.003	0.003	0.003	0.009	0.005	152.0

15	0.2	#ID/success	84/200	84/200	90/200	85/200	90/200	-
		runtime (s)	0.004	0.005	0.004	0.007	0.006	-
	0.5	#ID/success	84/200	84/200	101/200	87/200	104/200	-
		runtime (s)	0.004	0.004	0.005	0.01	0.01	-
	0.8	#ID/success	84/200	85/200	108/200	89/200	110/200	-
		runtime (s)	0.004	0.004	0.004	0.02	0.01	-
20	0.2	#ID/success	69/200	69/200	72/200	70/200	72/200	-
		runtime (s)	0.007	0.007	0.007	0.01	0.008	-
	0.5	#ID/success	69/200	72/200	86/200	78/200	90/200	-
		runtime (s)	0.007	0.007	0.007	0.03	0.01	-
	0.8	#ID/success	69/200	76/200	95/200	84/200	100/200	-
		runtime (s)	0.003	0.003	0.003	0.14	0.02	-
30	0.2	#ID/success	45/200	45/200	47/200	45/200	47/200	-
		runtime (s)	0.03	0.03	0.03	0.05	0.03	-
	0.5	#ID/success	45/200	48/200	61/200	51/200	62/200	-
		runtime (s)	0.03	0.03	0.03	0.17	0.05	-
	0.8	#ID/success	45/200	48/200	65/200	51/200	69/200	-
		runtime (s)	0.03	0.03	0.03	0.31	0.07	-
40	0.2	#ID/success	20/200	21/200	24/200	22/200	25/200	-
		runtime (s)	0.03	0.03	0.03	0.06	0.04	-
	0.5	#ID/success	20/200	22/200	30/200	22/200	30/200	-
		runtime (s)	0.03	0.03	0.03	0.19	0.06	-
	0.8	#ID/success	20/200	22/200	41/200	28/200	44/200	-
		runtime (s)	0.03	0.03	0.03	0.70	0.09	-

50	0.2	#ID/success	15/200	15/200	19/200	15/200	19/200	-
		runtime (s)	0.06	0.06	0.06	0.24	0.08	-
	0.5	#ID/success	15/200	19/200	31/200	19/199	33/200	-
		runtime (s)	0.06	0.07	0.06	2.68	0.21	-
	0.8	#ID/success	15/200	20/200	37/200	25/200	41/200	-
		runtime (s)	0.06	0.07	0.06	4.08	0.37	-

Table C.1: We record the number of causal effects deemed identifiable for each method under causal graphs with varying numbers of nodes (n) and CSIs (r). We also record the number of runs that complete within the 5-minute timeout (labeled as “success” in the table). For runs that exceed the timeout, we set the runtime to 5 minutes when computing the average.

APPENDIX D

Appendix for Chapter 5

D.1 Proofs

Proof of Proposition 6. The result follows from Definition 3 and Definition 10. $\square$

Proof of Proposition 7. By Proposition 6, it suffices to show that the output of the AC is $\langle G, \mathbf{V}, \mathcal{C} \rangle$ -invariant. If we replace all the expressions of nodes on the invariant-cut with their projections, it is guaranteed that the output expression of the AC (with projections) only involves variables $\mathbf{V}$, which is determined by $\Pr(\mathbf{V})$. $\square$

Proof of Theorem 8. First note that the completeness proof of the ID algorithm [SP06] assumes all variables are binary. Hence, the if direction of the theorem must hold.

Before proving the only-if direction, we briefly review the IDENTIFY algorithm in [TP03, HV06], which is both sound and complete. The algorithm starts by omitting all variables that are not ancestors of $\mathbf{Y}$ in the mutilated graph H and decomposing the resulting graph into c-components $\mathcal{S} = \{\mathbf{S}_1, \dots, \mathbf{S}_k\}$.¹ The goal is then to identify the c-factor for each c-component $\mathbf{S}_i$. This is implemented through a recursive procedure that applies two operations to a variable set $\mathbf{Z} = \mathbf{V}$: (i) set $\mathbf{Z} = An(\mathbf{S}_i)_{G_{\mathbf{Z}}}$, the ancestors of $\mathbf{S}_i$ in the subgraph $G_{\mathbf{Z}}$; and (ii) replace $\mathbf{Z}$ with a maximal c-component of $G_{\mathbf{Z}}$ that contains $\mathbf{S}_i$. The c-component $\mathbf{S}_i$ (and its c-factor) is identifiable iff $\mathbf{Z} = \mathbf{S}_i$ when the recursion terminates, and the causal effect is identifiable iff all the c-components in $\mathcal{S}$ are identifiable.

¹ H is obtained from the original causal graph G by removing treatment variables $\mathbf{X}$.

One key observation is that both operations are captured by the steps in Definition 12. Operation (ii) is equivalent to recursively pruning leaf nodes that are not in $\mathbf{S}_i$, and operation (ii) is identical to step (1) of Definition 12. Hence, a c-component $\mathbf{S}_i$ is identifiable only if it is $\mathbf{V}$ -computable, and we can always find a projection for $\mathbf{S}_i$ if it can be identified by IDENTIFY. We can then construct an AC based on the expression $\sum_{\mathbf{V} \setminus (\mathbf{X} \cup \mathbf{Y})} \prod_{\mathbf{S}_i \in \mathcal{S}} Q[\mathbf{S}_i]$, where $Q[\mathbf{S}_i]$ denotes the c-factor for the c-component $\mathbf{S}_i$. Note that the AC always computes the causal effect under all models. Moreover, if the causal effect is identifiable, we can always find projections for the AC nodes corresponding to the $Q[\mathbf{S}_i]$'s to form an invariant-cut. $\square$

Proposition 22. *Consider the causal graph G in Figure 5.5(b) where $\mathbf{V} = \{X, Y\}$. The causal effect $\Pr_x(y)$ is unidentifiable wrt $\langle G, \mathbf{V}, \Pr(\mathbf{V}) = \Pr^*(\mathbf{V}) \rangle$ for all $\Pr^*(\mathbf{V}) > 0$.*

WLG, to show that the causal effect $\Pr_x(y)$ is unidentifiable under any $\Pr^*(X, Y) > 0$, we construct two parameterizations Θ^1 and Θ^2 that agree on any $\Pr^*(X, Y)$ but disagree on $\Pr_{x_1}(y_1)$, where $\{x_1, \dots, x_n\}$ are the states of X and $\{y_1, \dots, y_m\}$ are the states of Y . Moreover, we assume that the hidden variables U is binary and has two states u_1, u_2 . We start by constructing a parameterization for G that satisfy certain conditions.

Lemma 8. *There exists a parameterization Θ for G that induces $\Pr^*(X, Y)$ and in which (i) all CPTs are strictly positive; and (ii) $\theta_{x|u_1} \neq \theta_{x|u_2}$ for all states x of X .*

Proof. We first construct a parameterization Θ for the subgraph $X \rightarrow Y$ that induces exactly $\Pr^*(X, Y)$. In particular, let $\theta_x = \Pr^*(x)$ and $\theta_{y|x} = \Pr^*(y|x)$ for all states x and y . Note that all values in Θ are in $(0, 1)$ since $\Pr^*(X, Y) > 0$.

We next construct a parameterization Θ' for the subgraph $U \rightarrow X \rightarrow Y$ based on Θ . In particular, let $\epsilon \rightarrow 0^+$ be an arbitrarily small constant, we assign

$$\theta'_{u_1} = \theta'_{u_2} = 0.5, \quad \theta'_{x|u_1} = \theta_x + \epsilon, \quad \theta'_{x|u_2} = \theta_x - \epsilon, \quad \theta'_{y|x} = \theta_{y|x}$$

for all states x, y . Let $\Pr'$ be the distribution induced by Θ' , we show that $\Pr' = \Pr^*$ as

follows.

$$\begin{aligned}\Pr'(x, y) &= \theta'_{u_1} \theta'_{x|u_1} \theta'_{y|x} + \theta'_{u_2} \theta'_{x|u_2} \theta'_{y|x} = 0.5(\theta_x + \epsilon) \theta_{y|x} + 0.5(\theta_x - \epsilon) \theta_{y|x} \\ &= \theta_x \theta_{y|x} = \Pr^*(x, y).\end{aligned}$$

By construction, Θ' satisfies both conditions (i) & (ii).

We finally construct Θ'' based on Θ' by assigning the same CPTs for U and X but different CPTs for Y . In particular,

$$\theta''_{y|u_1, x} = \theta''_{y|u_2, x} = \theta'_{y|x}$$

Θ'' also induces the distribution $\Pr^*$ since $(Y \perp\!\!\!\perp U | X)$. □

Proof of Proposition 22. Let Θ^1 be parameterization that satisfies the conditions in Lemma 8, we next construct Θ^2 that also induces $\Pr^*(X, Y)$ but disagrees with Θ^1 on the causal effect $\Pr_{x_1}(y_1)$. We do so by assigning same CPTs for U and X , i.e., $\Theta_U^2 = \Theta_U^1$ and $\Theta_X^2 = \Theta_X^1$, but different CPTs for Y . Specifically, for each state $y \neq y_1$, we assign

$$\theta_{y|u_1, x}^2 = \begin{cases} \theta_{y|u_1, x}^1 + \epsilon & \text{if } y = y_1 \\ \theta_{y|u_1, x}^2 = \theta_{y|u_1, x}^1 - \frac{\epsilon}{m-1} & \text{if } y \neq y_1 \end{cases}$$

and

$$\theta_{y|u_2, x}^2 = \begin{cases} \theta_{y|u_2, x}^1 - \frac{\Pr^1(u_1, x_1)}{\Pr^1(u_2, x_1)} & \text{if } y = y_1 \\ \theta_{y|u_2, x}^1 + \frac{\Pr^1(u_1, x_1)}{\Pr^1(u_2, x_1)} \frac{\epsilon}{(m-1)} & \text{if } y \neq y_1 \end{cases}$$

where $\Pr^1$ denotes the distribution induced by Θ^1 .

We first prove show that Θ^2 induces a distribution $\Pr^2$ where $\Pr^2(X, Y) = \Pr^1(X, Y) =$

$\Pr^*(X, Y)$. We first consider the instantiations that contains y_1 ,

$$\begin{aligned}
\Pr^2(x, y_1) &= \theta_{u_1}^2 \theta_{x|u_1}^2 \theta_{y_1|u_1, x}^2 + \theta_{u_2}^2 \theta_{x|u_2}^2 \theta_{y_1|u_2, x}^2 \\
&= \theta_{u_1}^1 \theta_{x|u_1}^1 (\theta_{y_1|u_1, x}^1 + \epsilon) + \theta_{u_2}^1 \theta_{x|u_2}^1 (\theta_{y_1|u_2, x}^1 - \frac{\Pr^1(u_1, x)}{\Pr^1(u_2, x)} \epsilon) \\
&= \Pr^1(x, y_1).
\end{aligned}$$

The last step is due to the fact that $\theta_{u_1}^1 \theta_{x|u_1}^1 = \Pr(u_1, x)$. We next consider the case when $y \neq y_1$.

$$\begin{aligned}
\Pr^2(x, y) &= \theta_{u_1}^2 \theta_{x|u_1}^2 \theta_{y|u_1, x}^2 + \theta_{u_2}^2 \theta_{x|u_2}^2 \theta_{y|u_2, x}^2 \\
&= \theta_{u_1}^1 \theta_{x|u_1}^1 (\theta_{y|u_1, x}^1 - \frac{\epsilon}{m-1}) + \theta_{u_2}^1 \theta_{x|u_2}^1 (\theta_{y|u_2, x}^1 + \frac{\Pr^1(u_1, x)}{\Pr^1(u_2, x)} \frac{\epsilon}{(m-1)}) \\
&= \Pr^1(x, y).
\end{aligned}$$

Together with the previous result, we have $\Pr^2(X, Y) = \Pr^*(X, Y)$. We are left to show that Θ^1 and Θ^2 induce different causal effect, i.e., $\Pr_{x_1}^1(y_1) \neq \Pr_{x_1}^2(y_1)$, which can be proved as follows.

$$\begin{aligned}
\Pr_{x_1}^2(y_1) &= \theta_{u_1}^2 \theta_{y_1|u_1, x_1}^2 + \theta_{u_2}^2 \theta_{y_1|u_2, x_1}^2 \\
&= \theta_{u_1}^1 (\theta_{y_1|u_1, x_1}^1 + \epsilon) + \theta_{u_2}^1 (\theta_{y_1|u_2, x_1}^1 - \frac{\Pr^1(u_1, x_1)}{\Pr^1(u_2, x_1)} \epsilon) \\
&= \Pr_{x_1}^1(y_1) + \theta_{u_1}^1 \epsilon - \theta_{u_2}^1 \frac{\Pr^1(u_1, x_1)}{\Pr^1(u_2, x_1)} \epsilon \\
&= \Pr_{x_1}^1(y_1) + \frac{\Pr^1(u_1)(\Pr^1(x_1|u_2) - \Pr^1(x_1|u_1))}{\Pr^1(x_1|u_2)} \epsilon \\
&\neq \Pr_{x_1}^1(y_1).
\end{aligned}$$

since $\Pr^1$ satisfies Conditions (i) & (ii) in Lemma 8. □

APPENDIX E

Appendix for Chapter 6

E.1 Variable Elimination Under Functional Dependencies

We review some key results in [Dar20] that enable the exploitation of functional dependencies. For simplicity, we use $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$ to denote sets of factors, where each set is interpreted as a product of its factors. For example, the set $\mathcal{F}$ can be interpreted as $\prod_{f \in \mathcal{F}} f$.

The first two theorems are general results for computing marginals [ZP96, Dec96]. The first theorem states that variables can be summed out in any order.

Theorem 17. $\sum_{\mathbf{XY}} f = \sum_{\mathbf{X}} \sum_{\mathbf{Y}} f = \sum_{\mathbf{Y}} \sum_{\mathbf{X}} f$.

The second theorem allows us to pull out factors from sums.

Theorem 18. *If variables $\mathbf{X}$ appear in factor f but not in factor g , then $\sum_{\mathbf{X}} f \cdot g = g \sum_{\mathbf{X}} f$.*

When functional dependencies are available, one can leverage the mechanisms to sum out variables earlier as shown in [Dar20].

Theorem 19 ([Dar20]). *Let f be a mechanism for variable X . If $f \in \mathcal{G}$ and $f \in \mathcal{H}$, then $\mathcal{G} \cdot \mathcal{H} = \mathcal{G} \sum_X \mathcal{H}$.*

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

Corollary 9 ([Dar20]). *If f is a mechanism for X , $f \in \mathcal{G}$ and $f \in \mathcal{H}$, then $\sum_X \mathcal{G} \cdot \mathcal{H} = (\sum_X \mathcal{G}) (\sum_X \mathcal{H})$.*

That is, if a mechanism for X appears in both parts of a product, we can sum out variable X from the product by independently summing it out from each part. Corollary 9 may appear unusable as it is predicated on multiple occurrences of a mechanism whereas the factors of a Bayesian network contain a single mechanism for each variable. This is where the second theorem comes in: *replicating* (i.e., duplicating) mechanisms in a product does not change the product value.

Theorem 20 ([Dar20]). *For mechanism f , if $f \in \mathcal{G}$, then $f \cdot \mathcal{G} = \mathcal{G}$.*

Consider the factor $\alpha = \sum_X f(XY)g(XZ)h(XW)$. The classical method has to multiply factors f , g and h before summing out variable X , therefore constructing a factor over four variables $XYZW$. However, if factor f is a mechanism for variable X , then we can replicate it by Theorem 20: $\alpha = f(XY)g(XZ)f(XY)h(XW)$. Corollary 9 then gives $\alpha = [\sum_X f(XY)g(XZ)][\sum_X f(XY)h(XW)]$. Hence, we can now compute factor α without having to construct any factor over more than three variables. Moreover, we were able to do this without needing to know the function represented by factor $f(XY)$: we only needed to know that this factor represents a function from Y to X . As shown in [Dar20], this technique can lead to exponential savings that are attained without needing to know the identity of mechanisms which is a major departure from earlier techniques.

E.2 Proofs

Our proofs are ordered slightly differently than the corresponding results in the main dissertation as we need some results when proving others. In the upcoming proofs, we will use f_X to denote a mechanism for variable X . We will say that a leaf node i in a jointree *contains* variable X iff X appears in a factor that is hosted at leaf node i . We will also use $\mathbf{vars}(i)$ to denote the variables of factors $\mathcal{F}_i$. For a jointree edge (i, j) , we will use $\mathbf{vars}(i, j)$ to denote the union of $\mathbf{vars}(k)$ for every leaf node k on the i -side of the edge.

Proof of Theorem 11

In this proof, we will assume we have a mapping from jointree edges (i, j) to sets $\mathbf{S}'_{ij} \subseteq \mathbf{S}_{ij}$ that satisfy conditions (a,b,c) of Definition 14 (that is, these sets satisfy only the first part of this definition but not the second part so they are not necessarily thinned separators). We will also say that jointree nodes i and j are $(X, \mathbf{S}')$ -connected iff $i = j$ or variable X appears in every set $\mathbf{S}'$ that is attached to an edge on the path between i and j . We will first state and prove two lemmas which we need for the proof of this theorem.

Lemma 9. *If $X \in \mathbf{S}'_{ij}$, then node i is $(X, \mathbf{S}')$ -connected to some leaf node on the i -side of edge (i, j) which contains variable X and node j is $(X, \mathbf{S}')$ -connected to some leaf node on the j -side of the edge which also contains X .*

Proof. Suppose node i is not $(X, \mathbf{S}')$ -connected to some leaf node on the i -side of edge (i, j) which contains X . Consider a longest path $i, \dots, r, l$ on the i -side of the edge such that X appears in $\mathbf{S}'$ for each edge on the path. If l is a leaf node, then it must contain variable X by definition of separators, which is a contradiction. Suppose now that l is not a leaf node. Then $X \notin \mathbf{S}'_{lk}$ for $k \neq r$, otherwise the path would not be longest. This contradicts the assumed condition (b) of Definition 14 on sets $\mathbf{S}'$. Hence, node i must be $(X, \mathbf{S}')$ -connected to some leaf node that contains X on the i -side of edge (i, j) . We can similarly show the second part of the lemma. $\square$

Lemma 10. *Suppose sets $\mathbf{S}'_{ij}$ were obtained by exhausting thinning rules on separators $\mathbf{S}_{ij}$. If node k is $(X, \mathbf{S}')$ -connected to some leaf node that contains X , then node k is $(X, \mathbf{S}')$ -connected to exactly one leaf node that hosts f_X .*

Proof. Suppose node k is $(X, \mathbf{S}')$ -connected to some leaf node that contains X . Then node k cannot be $(X, \mathbf{S}')$ -connected to more than one leaf node hosting f_X ; otherwise, Rule (a) will apply. We next show that node k must be $(X, \mathbf{S}')$ -connected to at least one mechanism f_X . We will show this by induction on the number of rule applications. We will use $\mathbf{S}^n$ to

denote the state of separators after the n^{th} rule application. For the base case (before any thinning rules are applied), node k must be (X, S^0) -connected to some f_X by the definition of separators in a jointree. For the inductive step, suppose node k is $(X, \mathbf{S}^n)$ -connected to some leaf node that contains X only if node k is $(X, \mathbf{S}^n)$ -connected to at least one leaf node that hosts f_X . Suppose now that node k is $(X, \mathbf{S}^{n+1})$ -connected to some leaf node that contains X . We will next show that node k must be $(X, \mathbf{S}^{n+1})$ -connected to at least one mechanism f_X .

First, node k must be $(X, \mathbf{S}^n)$ -connected to some leaf node that contains X . Hence, by the induction hypothesis, node k must be $(X, \mathbf{S}^n)$ -connected to at least one mechanism f_X . We will consider an edge (i, j) such that $X \in \mathbf{S}_{ij}^n$ and $X \notin \mathbf{S}_{ij}^{n+1}$ (such an edge must exist) and do a case analysis on which rule applied to this edge.

Case: Rule (a). We will show next that node k must be $(X, \mathbf{S}^{n+1})$ -connected to at least one mechanism f_X while assuming that node k is on the i -side of edge (i, j) . A similar argument will show the same if node k is on the j -side of the edge. Suppose node k is on the i -side of edge (i, j) . If node k is $(X, \mathbf{S}^n)$ -connected to some f_X on the i -side, then node k is $(X, \mathbf{S}^{n+1})$ -connected to the same f_X on the i -side. If node k is $(X, \mathbf{S}^n)$ -connected to some f_X on the j -side, then node k is $(X, \mathbf{S}^{n+1})$ -connected to node i . By the definition of Rule (a), node i must be $(X, \mathbf{S}^{n+1})$ -connected to some f_X on the i -side. Therefore, node k must be $(X, \mathbf{S}^{n+1})$ -connected to the same f_X on the i -side.

Case: Rule (b). Then none of the neighboring separators except $\mathbf{S}_{ij}^n$ contains X . If node k is on the i -side, then node k must be $(X, \mathbf{S}^n)$ to some f_X on the i -side and therefore must be $(X, \mathbf{S}^{n+1})$ to the same f_X on the i -side. The same argument applies if node k is on the j -side.

Case: Rule (c). Symmetric to the previous case. □

We need the following lemma which states the same property of Lemma 10 except under different conditions. The proof of this lemma uses the notion of “a closest leaf k to

node i which hosts mechanism f_X .” This is a leaf node that hosts f_X where the path $i = p_0, p_1, \dots, p_{n-1}, p_n = k$ has a minimal number of sets $\mathbf{S}_{p_i p_{i+1}}^*$ that do not contain X .

Lemma 11. *Consider thinned separators $\mathbf{S}^*$ according to Definition 14 and let X be a functional variable. If node i is $(X, \mathbf{S}^*)$ -connected to some leaf node that contains X , then node i is $(X, \mathbf{S}^*)$ -connected to exactly one leaf node that hosts mechanism f_X .*

Proof. Suppose node i is $(X, \mathbf{S}^*)$ -connected to some leaf node that contains X . Node i cannot be $(X, \mathbf{S}^*)$ -connected to two different leaves that host f_X as this would violate condition (a) of Definition 14. Suppose now that node i is not $(X, \mathbf{S}^*)$ -connected to any mechanism f_X . We will next show a contradiction. Consider the path $i = p_0, p_1, \dots, p_{n-1}, p_n = k$ where k is a closest leaf to node i which hosts mechanism f_X . We claim that adding X to all sets $\mathbf{S}_{p_i p_{i+1}}^*$ which do not contain X on the path results in a jointree thinning that still satisfies conditions (a,b,c) of Definition 14. This would be a contradiction as it implies there is a superset of thinning $\mathbf{S}^*$ that satisfies these conditions.

To show the above claim, note that conditions (b,c) will immediately continue to be satisfied if we add variables to sets $\mathbf{S}_{p_i p_{i+1}}^*$. We next show that condition (a) will continue to be satisfied as well.

Suppose we are adding X to sets $\mathbf{S}_{p_i p_{i+1}}^*$ which do not contain variable X using the given order of these sets. Let $\mathbf{S}'$ be the separators after adding X to $\mathbf{S}_{p_i p_{i+1}}^*$ when the first violation to condition (a) takes place. Then node p_i will be $(X, \mathbf{S}')$ -connected to a leaf node p'_i that hosts mechanism f_X on the p_i -side of edge (p_i, p_{i+1}) . This contradicts with the definition of k as p'_i will be closer to i than k . $\square$

Proof of Theorem 11. We first show the if condition. Note that conditions (a,b,c) of Definition 14 hold when we exhaust thinning rules. Suppose sets $\mathbf{S}'_{ij}$ were obtained by exhausting thinning rules on separators $\mathbf{S}_{ij}$ (and hence satisfy the three conditions). We just need to prove that no supersets of $\mathbf{S}'_{ij}$ satisfy these conditions. Suppose by contradiction such supersets $\mathbf{S}''_{ij}$ exist and consider an edge (i, j) such that $X \in \mathbf{S}''_{ij}$ and $X \notin \mathbf{S}'_{ij}$. By Lemma 9

since sets $\mathbf{S}_{ij}''$ satisfy the three conditions, node i must $(X, \mathbf{S}'')$ -connect to some leaf node k on the i -side which contains X . Since leaf node k contains X , it $(X, \mathbf{S}')$ -connects to itself. By Lemma 10, the leaf node k is $(X, \mathbf{S}')$ -connected to some f_X on the i -side since $X \notin \mathbf{S}_{ij}'$. Hence, leaf node k is also $(X, \mathbf{S}'')$ -connected to the same f_X on the i -side since sets $\mathbf{S}_{ij}''$ are supersets of $\mathbf{S}_{ij}'$. We have shown that node i is $(X, \mathbf{S}'')$ -connected to leaf node k which is $(X, \mathbf{S}'')$ -connected to some f_X on the i -side, therefore node i is $(X, \mathbf{S}'')$ -connected to some f_X on the i -side. By a similar argument, node j is $(X, \mathbf{S}'')$ -connected to some f_X on the j -side. However, this implies that the supersets $\mathbf{S}_{ij}''$ violate condition (a), which is a contradiction. Hence, the supersets $\mathbf{S}_{ij}''$ cannot exist.

We now show the only-if direction. Given a jointree thinning $\mathbf{S}^*$ that satisfies Definition 14, we will show how to construct a thinning sequence that produces it. A key observation is that rule applications for a variable X are independent of rule applications for a variable $Y \neq X$; that is, we can always rearrange a thinning sequence so rules that apply to the same variable are consecutive. Hence, we will construct a thinning sequence that produces $\mathbf{S}^*$ by constructing a set of rule applications for each variable and then paste them together.

To construct the rule applications for variable X , we start with some leaf node l which hosts mechanism f_X and then traverse nodes away from l . Suppose we are visiting node i now which has a neighbor j that has not been visited. If $X \notin \mathbf{S}_{ij}^*$ and $X \in \mathbf{S}_{ij}$, we consider two cases. If $X \notin \mathbf{S}_{jk}^*$ for all k , we add $R_b(i, j, X)$. Otherwise, we add $R_a(i, j, X)$. Every variable that has been thinned will now be accounted for by a rule. Moreover, these rules will be applicable in the reverse order in which they have been constructed. Each $R_a(i, j, X)$ will be applicable by Lemmas 9 and 11. Each $R_b(i, j, X)$ will be applicable by definition. $\square$

Proof of Theorem 10

Our proof starts with the correctness of the message passing algorithm using classical separators and then shows that the algorithm continues to be sound after we apply a thinning

rule to remove a variable from some separator (recall that every jointree thinning can be obtained by a sequence of thinning rules). We will use $\mathbf{S}'$ to indicate the state of separators after some rule applications.

Our proof uses the following variant on Definition 13. We will say that a jointree node i is *strongly* $(X, \mathbf{S}')$ -connected to a factor f iff i hosts f or $\text{vars}(f)$ appears in every separator $\mathbf{S}'_{kl}$ on the path between node i and some leaf node j that hosts f . Similarly, we will say that jointree nodes i and j are *strongly* $(X, \mathbf{S}')$ -connected iff $i = j$ or variables $\text{vars}(f_X)$ appear in every separator $\mathbf{S}'_{kl}$ on the path between i and j .

The thinning rules for distinct variables do not interact with one another. Hence, we will assume in this proof that all thinning rules are applied according to a reverse topological ordering π of the variables in the underlying DAG.

We will use two lemmas in this proof. The first says that X -connection (Definition 13) and strong X -connection (defined above) are equivalent when applying thinning rules according to reverse topological ordering π .

Lemma 12. *Suppose $\mathbf{S}'$ is the state of separators after applying thinning rules to variables that do not follow variable X in order π . For jointree edge (i, j) , node i is $(X, \mathbf{S}')$ -connected to some f_X on the i -side of the edge and node j is $(X, \mathbf{S}')$ -connected to some f_X on the j -side of the edge iff i is strongly $(X, \mathbf{S}')$ -connected to some f_X on the i -side and j is strongly $(X, \mathbf{S}')$ -connected to some f_X on the j -side.*

Proof. The if part follows from the fact that $X \in \text{vars}(f_X)$. We next show the only-if part. Suppose node i is $(X, \mathbf{S}')$ -connected to node l that hosts f_X on the i -side and node j is $(X, \mathbf{S}')$ -connected to node r that hosts f_X on the j -side. By the property of classical separators, $\text{vars}(f_X) \subseteq \mathbf{S}_{xy}$ for all edges (x, y) on the path between l and r . Since π is a reverse topological ordering of the variables, none of the parents of X ($\text{vars}(f_X) \setminus \{X\}$) are thinned from the separators. Therefore, $\text{vars}(f_X) \setminus \{X\} \subseteq \mathbf{S}'_{xy}$ for all edges (x, y) on the path between l and r . Since node i is $(X, \mathbf{S}')$ -connected to l and node j is $(X, \mathbf{S}')$ -

connected to r , we conclude that node i is strongly $(X, \mathbf{S}')$ -connected to l and node j is strongly $(X, \mathbf{S}')$ -connected to r . $\square$

The second lemma extends Theorem 19 to a more general setting. For factors f and $\mathcal{F}$, we will write $f \in \mathcal{F}$ to mean that $\mathcal{F} = f \cdot g$ for some factor g .

Lemma 13. *Consider factor $\mathcal{G} = \sum_{\mathbf{S}_1} \gamma_1 \cdots \sum_{\mathbf{S}_k} \gamma_k \cdot \mathcal{H}$ where $\gamma_1, \dots, \gamma_k$ are arbitrary factors, $f_X \in \mathcal{H}$ and $\text{vars}(f_X) \subseteq \mathbf{S}_1, \dots, \text{vars}(f_X) \subseteq \mathbf{S}_k$ for some mechanism f_X of variable X . If $f_X \in \mathcal{F}$, then $\mathcal{F} \cdot \mathcal{G} = \mathcal{F} \cdot \mathcal{G}'$ where $\mathcal{G}' = \sum_{\mathbf{S}_1} \gamma_1 \cdots \sum_{\mathbf{S}_k} \gamma_k \cdot \sum_X \mathcal{H}$.*

Proof. Suppose $f_X \in \mathcal{F}$. Then $\mathcal{F} = f_X \cdot \mathcal{F}'$ for some factor $\mathcal{F}'$. Moreover, $\mathcal{F} \cdot \mathcal{G}$ equals to

$$\begin{aligned}
& \mathcal{F}' \cdot f_X \cdot \sum_{\mathbf{S}_1} \gamma_1 \cdots \sum_{\mathbf{S}_k} \gamma_k \cdot \mathcal{H} \\
&= \mathcal{F}' \cdot \sum_{\mathbf{S}_1} \gamma_1 \cdots \sum_{\mathbf{S}_k} \gamma_k \cdot f_X \cdot \mathcal{H} \\
&= \mathcal{F}' \cdot \sum_{\mathbf{S}_1} \gamma_1 \cdots \sum_{\mathbf{S}_k} \gamma_k \cdot f_X \cdot \sum_X \mathcal{H} \quad (\text{by Theorem 19}) \\
&= \mathcal{F}' \cdot f_X \cdot \sum_{\mathbf{S}_1} \gamma_1 \cdots \sum_{\mathbf{S}_k} \gamma_k \cdot \sum_X \mathcal{H} \\
&= \mathcal{F} \cdot \mathcal{G}'. \quad \square
\end{aligned}$$

Proof of Theorem 10. We are now ready for the soundness proof. For jointree edge (i, j) , let $\mathcal{M}'_{ij}$ and $\mathcal{M}'_{ji}$ denote the messages between i and j under separators $\mathbf{S}'$. We will next show $Pr(\mathbf{S}'_{ij}) = \mathcal{M}'_{ij} \mathcal{M}'_{ji}$ for all edges (i, j) by induction on rule applications. For each rule application, we will use $\mathbf{S}$ to denote the separators before thinning by the rule and $\mathbf{S}'$ to denote the separators after thinning by the rule. Initially, $Pr(\mathbf{S}_{ij}) = \mathcal{M}_{ij} \mathcal{M}_{ji}$ for all edges (i, j) by Theorem 9. We next show that this equality holds after each rule application. We consider three cases, one for each rule type.

(1) Rule (a) is applied to edge (i, j) : $X \in \mathbf{S}_{ij}$ and $X \notin \mathbf{S}'_{ij}$. By definition of Rule (a) and Lemma 12, node i is strongly $(X, \mathbf{S})$ -connected to some mechanism f_X hosted at leaf node l

on the i -side and node j is strongly $(X, \mathbf{S})$ -connected to some mechanism f_X hosted at leaf node r on the j -side.

First, we have $\mathcal{M}'_{ij} \cdot \mathcal{M}'_{ji} = \mathcal{M}_{ij} \cdot \mathcal{M}_{ji}$ by Corollary 9. Consider now any edge (k, z) on the path $l \dots k - z \dots r$ between leaf nodes l and r and suppose WLG that edge (i, j) is on the subpath $z \dots r$. Using Lemma 13 with $\mathcal{F} = \mathcal{M}_{kz}$, $\mathcal{G} = \mathcal{M}_{zk}$ and $\mathcal{H} = \mathcal{M}_{ji}$, we get $\mathcal{M}_{kz} \cdot \mathcal{M}_{zk} = \mathcal{M}'_{kz} \cdot \mathcal{M}'_{zk}$. That is, removing X from the separator of edge (i, j) does not affect the product of messages for edge (k, z) . Finally, consider any edge (k, z) that is not on the path between leaf nodes l and r . Let t be the node on this path which is closest to edge (k, z) . Let l', r', u be the neighbors of t that are closest to l , r and edge (k, z) , respectively. To show $\mathcal{M}_{kz} \cdot \mathcal{M}_{zk} = \mathcal{M}'_{kz} \cdot \mathcal{M}'_{zk}$, it suffices to show $\mathcal{M}'_{tu} = \mathcal{M}_{tu}$. WLG, suppose node j is closer to t than node i . Since $\mathcal{M}_{tu} = \sum_{\mathbf{S}_{tu}} \gamma \cdot \mathcal{M}_{l't} \cdot \mathcal{M}_{r't}$, where γ denotes the product of other invariant messages, we can use Lemma 13 again with $\mathcal{F} = \mathcal{M}_{l't}$, $\mathcal{G} = \mathcal{M}_{r't}$ and $\mathcal{H} = \mathcal{M}_{ij}$ to get $\mathcal{M}'_{l't} \cdot \mathcal{M}'_{r't} = \mathcal{M}_{l't} \cdot \mathcal{M}_{r't}$. Hence, applying Rule (a) preserves the product of messages for all jointree edges.

(2) Rule (b) is applied to edge (i, j) : $X \in \mathbf{S}_{ij}$ and $X \notin \mathbf{S}'_{ij}$. By definition of Rule (b), node i is not a leaf and $X \notin \mathbf{S}_{ki}$ for $k \neq j$. Then $\mathcal{M}'_{ij} = \sum_X \mathcal{M}_{ij} = \mathcal{M}_{ij}$ since messages $\mathcal{M}_{ki}$ do not contain X . Moreover, $Pr(\mathbf{S}'_{ij}) = \sum_X Pr(\mathbf{S}_{ij}) = \sum_X \mathcal{M}_{ij} \cdot \mathcal{M}_{ji} = \mathcal{M}_{ij} \cdot (\sum_X \mathcal{M}_{ji}) = \mathcal{M}'_{ij} \cdot \mathcal{M}'_{ji}$. We next consider edges other than (i, j) .

Since $\mathcal{M}'_{ij} = \mathcal{M}_{ij}$, all messages outgoing from node j are invariant. Hence, the product of messages is invariant for any edge on the j -side of edge (i, j) . We next show that all message outgoing from i to neighbors $k \neq j$ are also invariant. This shows that the product

of messages is also invariant for all edges on the k -side of any edge (k, i) .

$$\begin{aligned}
\mathcal{M}_{ik} &= \sum_{\mathbf{S}_{ik}}^{\equiv} \mathcal{M}_{ji} \cdot \prod_{t \neq k, t \neq j} \mathcal{M}_{ti} \\
&= \sum_{\mathbf{S}_{ik}}^{\equiv} \left(\sum_X \mathcal{M}_{ji} \right) \cdot \prod_{t \neq k, t \neq j} \mathcal{M}_{ti} \\
&= \sum_{\mathbf{S}_{ik}}^{\equiv} \mathcal{M}'_{ji} \cdot \prod_{t \neq k, t \neq j} \mathcal{M}'_{ti} \\
&= \mathcal{M}'_{ik}.
\end{aligned}$$

The second step follows since $X \notin \mathbf{S}_{ki}$ and also $X \notin \mathbf{S}_{ti}$. The third step follows since messages $\mathcal{M}_{ti}$ are invariant to thinning variable X from separator $\mathbf{S}_{ij}$.

(3) Rule (c) is applied to edge (i, j) . Similar to case (2). $\square$

Statement and Proof of Theorem 21

The following result shows that we cannot improve Definition 14 of jointree thinnings based only on Theorem 19 in Appendix E.1, indicating the optimality of thinnings.

Theorem 21. *Consider a jointree thinning (Definition 14). If we remove any functional variable from a thinned separator, then Theorem 19 will no longer be sufficient to prove the soundness of the message-passing algorithm (Theorem 9).*

The proof of the theorem requires a lemma that provides a characterization of jointree thinnings, which is more suitable for verifying whether the removal of variables from classical separators leads to a valid thinning.

Lemma 14. *A mapping from each jointree edge (i, j) to variable set $\mathbf{S}_{ij}^*$ is a jointree thinning according to Definition 14 iff (1) for each non-functional variable X , $X \in \mathbf{S}_{ij}^*$ iff $X \in \mathbf{vars}(i, j) \cap \mathbf{vars}(j, i)$; (2) for each functional variable X : (a) if $X \in \mathbf{vars}(i)$ for a leaf node i , then i is X -connected to exactly one mechanism for X ; (b) if $X \in \mathbf{S}_{ij}^*$ for some*

non-leaf i , then $X \in \mathbf{S}_{ik}^*$ for some $k \neq j$.

Proof. A jointree thinning according to Definition 14 satisfies the two properties of the theorem by Definition 14 and Lemma 11. Suppose now the two properties hold for a mapping $\mathbf{S}^*$. We will show there exists a thinning sequence that produces $\mathbf{S}^*$ from the classical separators $\mathbf{S}$. We first note that no thinning rules can be applied to a mapping $\mathbf{S}^*$ that satisfies the two properties of the theorem. To apply thinning rules to separators $\mathbf{S}$, we consider each functional variable X , then locate all connected subtrees where X does not appear in any separator of a subtree. Consider one such connected subtree Γ and let $\mathcal{B} = \{\mathbf{S}_{u_1 t_1}^*, \dots, \mathbf{S}_{u_k t_k}^*\}$ be the boundary separators of the subtree. WLG, assume u_i are at the subtree boundary. Observe that each u_i can either be a leaf node that hosts a mechanism f_X or a non-leaf node such that $X \in \mathbf{S}_{u_i r_i}^*$ for $r_i \neq t_i$. Similar to the proof for Lemma 9, we can show that each u_i is $(X, \mathbf{S}^*)$ -connected to some f_X . Let $\{\mathbf{S}_{u_1 t_1}, \dots, \mathbf{S}_{u_k t_k}\}$ be the original classical separators of the jointree, then the separators $\mathbf{S}^*$ for the connected subtree Γ can be obtained by the following thinning sequence. We first apply thinning Rule (a) to all boundary separators $\mathbf{S}_{u_1 t_1}, \dots, \mathbf{S}_{u_k t_k}$ but one. Suppose we apply the rule to separators $\mathbf{S}_{u_2 t_2}, \dots, \mathbf{S}_{u_k t_k}$. This is sound since u_1 is $(X, \mathbf{S})$ -connected to all $u_2, \dots, u_k$ by the property of classical separators. Starting from these thinned boundary separators, we can then thin X from all separators in subtree Γ using Rules (b,c). $\square$

Proof of Theorem 21. Suppose $\mathbf{S}^*$ is a jointree thinning according to Definition 14 and let X be a functional variable such that $X \in \mathbf{S}_{ij}^*$ for some edge (i, j) . By Lemma 14, we must have some leaf node k with $X \in \text{vars}(i)$ that is X -connected to exactly one mechanism for X , f_X , through edge (i, j) . WLG, suppose leaf k is on the i -side of the edge and mechanism f_X is on the j -side of the edge. Suppose further that we remove variable X from $\mathbf{S}_{ij}^*$ leading to new separators $\mathbf{S}'$. This will lead to a violation of Condition (2a) in Lemma 14. In particular, leaf node k will no longer be X -connected to any mechanism for X . Now let $\mathcal{F}$ be the factors on the i -side of edge (i, j) , $\mathcal{G}$ be the factors on the j -side of the edge and m be the number

of mechanisms f_X in $\mathcal{F} \cup \mathcal{G}$. Let $\mathcal{M}^*$ denote messages computed using separators $\mathbf{S}^*$ and $\mathcal{M}'$ denote the messages computed using the separators $\mathbf{S}'$. The computation of messages $\mathcal{M}_{ij}^*$ and $\mathcal{M}_{ji}^*$ must involve at least $m - 1$ distinct sum-outs of X . This follows because the mechanisms f_X cannot be X -connected. Since $\mathcal{M}_{ij}' = \sum_X \mathcal{M}_{ij}^*$ and $\mathcal{M}_{ji}' = \sum_X \mathcal{M}_{ji}^*$, computing the product $\mathcal{M}_{ij}' \cdot \mathcal{M}_{ji}'$ involves $(m - 1) + 2 = m + 1$ distinct sum-outs of variable X . If $\Pr(\mathbf{S}_{ij}') = \sum_{\mathbf{S}_{ij}'} \mathcal{F} \cdot \mathcal{G} = \mathcal{M}_{ij}' \cdot \mathcal{M}_{ji}'$ then $\mathcal{M}_{ij}' \cdot \mathcal{M}_{ji}'$ is a factorization of $\sum_{\mathbf{S}_{ij}'} \mathcal{F} \cdot \mathcal{G}$ that involves $m + 1$ distinct sum-outs of variable X . However, since factors $\mathcal{F} \cup \mathcal{G}$ contain exactly m replicas of mechanism f_X , any factorization of $\sum_{\mathbf{S}_{ij}'} \mathcal{F} \cdot \mathcal{G}$ cannot include more than m distinct sum-outs of variable X that are based on Theorems 18 and 19. This follows because each sum-out of X based on Theorem 19 will consume a mechanism f_X and the sum-outs based on Theorem 18 do not consume mechanisms. Hence, the equality $\sum_{\mathbf{S}_{ij}'} \mathcal{F} \cdot \mathcal{G} = \mathcal{M}_{ij}' \cdot \mathcal{M}_{ji}'$ cannot be justified based only on these two theorems. That is, Theorem 19 will no longer be sufficient to imply the soundness of the message-passing algorithm as stated in Theorem 9. $\square$

Proof of Theorem 12

Our first lemma states that adding mechanisms to a replication never decreases its width.

Lemma 15. *Consider two replications $\mathcal{F}_1$ and $\mathcal{F}_2$ of some factors $\mathcal{F}$ where $\mathcal{F} \subseteq \mathcal{F}_1 \subseteq \mathcal{F}_2$. If the width of replication $\mathcal{F}_1$ is w , then the width of replication $\mathcal{F}_2$ is $\leq w$.*

Proof. Suppose replication $\mathcal{F}_1$ has width w . Then it must have a thinned jointree $\mathcal{T}_1$ of width w . We can turn $\mathcal{T}_1$ into a thinned jointree $\mathcal{T}_2$ for replication $\mathcal{F}_2$ by assigning more factors to leaf nodes in $\mathcal{T}_1$. In particular, for each replica $f_X \in \mathcal{F}_2 \setminus \mathcal{F}_1$, assign this replica to a leaf node in $\mathcal{T}_1$ which hosts a mechanism f_X . This guarantees that $\mathcal{T}_2$ will also have width w . Since replication $\mathcal{F}_2$ has a thinned jointree of width w , its width must be $\leq w$. $\square$

Our second lemma shows that including more X -mechanisms than X -feeding factors in a replication provides no additional benefit.

Lemma 16. *Consider a replication $\mathcal{F}$ that contains $n > 1$ mechanisms for variable X and m X -feeding factors where $n > m$. Let $\mathcal{F}'$ be the result of removing one mechanism for X from $\mathcal{F}$ (hence, $|\mathcal{F}'| = |\mathcal{F}| - 1$). The width of $\mathcal{F}'$ is no greater than the width of $\mathcal{F}$.*

Proof. Let $\mathcal{T}$ be a thinned jointree for $\mathcal{F}$. We can turn $\mathcal{T}$ into a thinned jointree $\mathcal{T}'$ for $\mathcal{F}'$ with no greater width as follows. Suppose there exists a leaf node in $\mathcal{T}$ that hosts two mechanisms f_X . We can then remove one of these mechanisms from the leaf without increasing the width. Suppose now that each leaf node in $\mathcal{T}$ hosts at most one mechanism f_X . The edges of $\mathcal{T}$ which contain variable X in their (thinned) separators form a set of connected subtrees. Each one of these subtrees will contain at most one mechanism f_X (otherwise thinning Rule (a) will apply). By the pigeonhole principle, at least one of these subtrees must contain a mechanism f_X but no X -feeding factors. We can show that removing this mechanism f_X from the subtree would not increase the width of resulting tree. Hence, if $\mathcal{F}$ has a thinned jointree of width w , then $\mathcal{F}'$ has a thinned jointree of no greater width. This implies that the width of $\mathcal{F}'$ is no greater than the width of $\mathcal{F}$. $\square$

Proof of Theorem 12. We first construct a new replication $\mathcal{G} = \mathcal{F} \cup \mathcal{F}'$. By Lemma 15, the width of $\mathcal{G}$ is no greater than the width of $\mathcal{F}'$. We then go through the functional variables in reverse-topological order (as in Algorithm 3). For each variable X , we compare if the X -mechanisms in $\mathcal{F}$ and $\mathcal{G}$ are equal. If so, we proceed to the next functional variable in the order. Otherwise, $\mathcal{G}$ contains more mechanisms than $\mathcal{F}$ and thus there are more X -mechanisms than X -feeding factors in $\mathcal{G}$. By Lemma 16, we can remove the excess X -mechanisms from $\mathcal{G}$ without increasing its width. By the end of this process, $\mathcal{G}$ is equal to the complete replication $\mathcal{F}$. Hence, the width of $\mathcal{F}$ is no greater than the width of $\mathcal{F}'$. $\square$

Proof of Proposition 8

In a complete replication, the number of mechanisms for a functional variable is upper-bounded by the total number of mechanisms of its children. Let X be a functional vari-

able and $f(X)$ be the number of mechanisms in the complete replication, then $f(X) = \sum_{C_i} f(C_i) \leq c \cdot \max_{C_i} f(C_i)$ where C_i are the children of X in the DAG. Hence, we can recursively bound the number of mechanisms for all the functional nodes in a functional chain. Since the longest functional chain has a length of k , we can recursively apply the above bound for at most k steps. When the recursion terminates, $\max_{C_i} f(C_i) = 1$ since all $f(C_i)$ will be non-functional variables. Therefore, $f(X) \leq c^k$ for each functional variable X in the replication. The inequality also holds for non-functional variables, $f(X) = 1 = c^0 \leq c^k$. Given a total of n nodes in the DAG, we have at most $O(nc^k)$ factors in the complete replication.

APPENDIX F

Appendix for Chapter 7

F.1 Proofs

Proof of Theorem 13. From [CD22], a thinned jointree can be obtained by applying a sequence of thinning rules to a base jointree. Let $\mathbf{Q} = \{Q_1, \dots, Q_T\}$ denote the T thinning rules being applied to the base jointree in order. We next construct a thinning sequence $\mathbf{Q}^t$ for the twin jointree, which leads to the thinned separators defined in the theorem.

We first define a *parallel thinning step* as simultaneously thinning a functional variable X from the base jointree and thinning $\{X, [X]\}$ from the twin jointree. Consider any thinning rule Q_i that thins X from a separator $\mathbf{S}_{ij}$, the parallel thinning on $\mathbf{S}^t$ is defined as follows:

- Suppose (i, j) is a duplicated edge, then we thin X from $\mathbf{S}_{ij}^t$ and $[X]$ from $\mathbf{S}_{[i][j]}^t$
- Suppose (i, j) is an invariant edge, then we thin $\{X, [X]\}$ from $\mathbf{S}_{ij}^t$

First, note that by construction, the theorem holds for all separators prior to the application of any thinning rule. Moreover, by the definition of parallel thinning, the theorem continues to hold after each parallel thinning step. It remains to show that the parallel thinnings on $\mathbf{S}^t$ are indeed valid.

Let $\mathbf{S}, \mathbf{S}^t$ denote the separators for the base and twin jointree during the parallel thinnings. Consider a parallel thinning step being applied to a duplicated edge, which, by definition, removes X from $\mathbf{S}_{ij}$, X from $\mathbf{S}_{ij}^t$, and $[X]$ from $\mathbf{S}_{[i][j]}^t$. Suppose the removal of X from $\mathbf{S}_{ij}$ is supported by the first thinning rule, i.e. the edge (i, j) is on the path between two leaf nodes,

call them l and r , both hosting the family of X , and every separator on that path contains X . We claim that the removal of X from $\mathbf{S}_{ij}^t$ and the removal of $[X]$ from $\mathbf{S}_{[i][j]}^t$ are both supported by the first thinning rule. By Algorithm 1, the leaf nodes $\{l, r\}$ host the family of X , and the leaf nodes $\{[l], [r]\}$ host the family of $[X]$ in the twin jointree. Moreover, by the inductive assumption on separators, X appears on every separator between l and r and $[X]$ appears on every separator between $[l]$ and $[r]$. This is based on an observation that the path from l to r can be divided into three sub-paths: a sub-path consisting of only duplicated edges ($l = p_1, \dots, p_s$), a sub-path consisting of only invariant edges ($p_s, \dots, p_m$), and a sub-path consisting of only duplicated edges ($p_m, \dots, p_n = r$), where $1 < s \leq m < n^1$. Given the path from l to r , we can then express the path from $[l]$ to $[r]$ as three sub-paths as well: a sub-path consisting of only duplicate edges ($[l] = [p_1], \dots, p_s$), a sub-path consisting of only invariant edges ($p_s, \dots, p_m$), and a sub-path consisting of only duplicate edges ($p_m, \dots, [p_n] = [r]$). For each duplicated edge (i, j) on the sub-path from l to p_s or on the sub-path from p_m to r , $X \in \mathbf{S}_{ij}$ iff $[X] \in \mathbf{S}_{[i][j]}$. For each invariant edge (i, j) on the sub-path from p_s to p_m , $X \in \mathbf{S}_{ij}$ iff $[X] \in \mathbf{S}_{ij}$.

Suppose the removal of X from $\mathbf{S}_{ij}$ is supported by the second thinning rule, i.e. no other separators $\mathbf{S}_{ik}$ contains X , or no other separators $\mathbf{S}_{kj}$ contains X . Then the removal of X from $\mathbf{S}_{ij}^t$ and the removal of $[X]$ from $\mathbf{S}_{[i][j]}^t$ are both supported by the second thinning rule due to the inductive assumption on separators.

Consider now a parallel thinning step being applied to an invariant edge which removes X from $\mathbf{S}_{ij}$ and $\{X, [X]\}$ from $\mathbf{S}_{ij}^t$. Suppose the removal of X from $\mathbf{S}_{ij}$ is supported by the first thinning rule, where the edge (i, j) is on the path between two leaf nodes l and r both hosting the family of X and every separator on that path contains X . Again, by the inductive assumption on separators, $[X]$ appears in all separators on the path between $[l]$ and $[r]$, which also includes the invariant edge (i, j) . Hence, $[X]$ can also be removed from $\mathbf{S}_{ij}^t$ using the first thinning rule. Suppose the removal of X from $\mathbf{S}_{ij}$ is supported by the

¹Note that we do not preclude the possibility of having no invariant edge on the path.

second thinning rule, then we can apply the second thinning rule to remove $\{X, [X]\}$ from $\mathbf{S}_{ij}^t$ due to the inductive assumption. $\square$

Proof of Theorem 14. We can extend Algorithm 4 to construct N -world jointrees by making $N - 1$ duplicates for each duplicated subtree, and construct the thinned separators for N -world thinned jointrees by extending Theorem 13. In particular, we assign to the i^{th} duplicated edge a separator that contains the i^{th} duplicates of the variables in the original separator, and to each invariant edge a separator containing all duplicates of the variables in the original separator. The width of the resulting N -world (thinned) jointrees is at most $N(w + 1) - 1$, where w is the width of the base jointree. Similar to the proof of Theorem 13, we can justify the validity of thinned separators through parallel thinning steps that remove each variable together with all its duplicates simultaneously. $\square$

REFERENCES

- [AP89] Stefan Arnborg and Andrzej Proskurowski. “Linear time algorithms for NP-hard problems restricted to partial k-trees.” *Discret. Appl. Math.*, **23**(1):11–24, 1989.
- [ASP05] Chen Avin, Ilya Shpitser, and Judea Pearl. “Identifiability of Path-Specific Effects.” In *IJCAI*, pp. 357–363. Professional Book Center, 2005.
- [BFG96] Craig Boutilier, Nir Friedman, Moisés Goldszmidt, and Daphne Koller. “Context-Specific Independence in Bayesian Networks.” In *UAI*, pp. 115–123. Morgan Kaufmann, 1996.
- [BP94a] Alexander Balke and Judea Pearl. “Counterfactual Probabilities: Computational Methods, Bounds and Applications.” In *UAI*, pp. 46–54. Morgan Kaufmann, 1994.
- [BP94b] Alexander Balke and Judea Pearl. “Probabilistic Evaluation of Counterfactual Queries.” In *AAAI*, pp. 230–237. AAAI Press / The MIT Press, 1994.
- [BP95] Alexander Balke and Judea Pearl. “Counterfactuals and Policy Analysis in Structural Models.” In *UAI*, pp. 11–18. Morgan Kaufmann, 1995.
- [CD07] Mark Chavira and Adnan Darwiche. “Compiling Bayesian Networks Using Variable Elimination.” In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2443–2449, 2007.
- [CD22] Yizuo Chen and Adnan Darwiche. “On the Definition and Computation of Causal Treewidth.” In *UAI, Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence*, 2022.
- [CD24] Yizuo Chen and Adnan Darwiche. “Identifying Causal Effects Under Functional Dependencies.” *Entropy*, **26**(12), 2024.
- [CD25a] Yizuo Chen and Adnan Darwiche. “Constrained Identifiability of Causal Effects.” *AAAI Workshop on Artificial Intelligence with Causal Techniques*, 2025.
- [CD25b] Yizuo Chen and Adnan Darwiche. “On the Granularity of Causal Effect Identifiability.” *arXiv*, **abs/2510.16703**, 2025.
- [CHD24] Yizuo Chen, Haiying Huang, and Adnan Darwiche. “Towards an effective practice of learning from data and knowledge.” *Int. J. Approx. Reason.*, **171**:109188, 2024.
- [CLB21] Juan D. Correa, Sanghack Lee, and Elias Bareinboim. “Nested Counterfactual Identification from Arbitrary Surrogate Experiments.” In *NeurIPS*, pp. 6856–6867, 2021.

- [Coo90] Gregory F. Cooper. “The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks.” *Artif. Intell.*, **42**(2-3):393–405, 1990.
- [Dar01] Adnan Darwiche. “Decomposable negation normal form.” *J. ACM*, **48**(4):608–647, 2001.
- [Dar02] Adnan Darwiche. “A Logical Approach to Factoring Belief Networks.” In *KR*, pp. 409–420. Morgan Kaufmann, 2002.
- [Dar03] Adnan Darwiche. “A differential approach to inference in Bayesian networks.” *J. ACM*, **50**(3):280–305, 2003.
- [Dar09] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.
- [Dar20] Adnan Darwiche. “An Advance on Variable Elimination with Applications to Tensor-Based Computation.” In *ECAI*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 2559–2568. IOS Press, 2020.
- [Dar21a] Adnan Darwiche. “Causal Inference with Tractable Circuits.” In *WHY Workshop, NeurIPS*, 2021. <https://arxiv.org/abs/2202.02891>.
- [Dar21b] Adnan Darwiche. “Tractable Boolean and Arithmetic Circuits.” In *Neuro-Symbolic Artificial Intelligence*, volume 342 of *Frontiers in Artificial Intelligence and Applications*, pp. 146–172. IOS Press, 2021.
- [Dat12] C. J. Date. *Database Design and Relational Theory - Normal Forms and All That Jazz*. O’Reilly, 2012.
- [Dec96] Rina Dechter. “Bucket Elimination: A unifying framework for probabilistic inference.” In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 211–219, 1996.
- [DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. “Maximum Likelihood from Incomplete Data via the EM Algorithm.” *Journal of the Royal Statistical Society. Series B (Methodological)*, **39**(1):1–38, 1977.
- [DMM17] Philip Dawid, Monica Musio, and Rossella Murtas. “The Probability of Causation.” *Law, Probability and Risk*, (16):163–179, 2017.
- [DRT25] Rina Dechter, Anna Raichev, Jin Tian, and Alexander Ihler. “Graph-based Complexity for Causal Effect by Empirical Plug-in.” In *AISTATS*, Proceedings of Machine Learning Research, pp. 4798–4806. PMLR, 2025.
- [ER59] Erdos, Paul and Renyi, Alfred. “On Random Graphs.” *Publicationes Mathematicae Debrecen*, **6**:290–297, 1959.

- [Eva18] Robin J. Evans. “Margins of discrete Bayesian networks.” *The Annals of Statistics*, **46**(6A):2623 – 2656, 2018.
- [GP88] Dan Geiger and Judea Pearl. “On the logic of causal models.” In *UAI*, pp. 3–14. North-Holland, 1988.
- [GP98] David Galles and Judea Pearl. “An axiomatic characterization of causal counterfactuals.” *Foundations of Science*, **3**(1):151–182, 1998.
- [GVP90] Dan Geiger, Thomas Verma, and Judea Pearl. “Identifying independence in bayesian networks.” *Networks*, **20**(5):507–534, 1990.
- [Hal00] Joseph Y. Halpern. “Axiomatizing causal reasoning.” *Journal of Artificial Intelligence Research*, **12**:317–337, 2000.
- [HCA23] David Huber, Yizuo Chen, Alessandro Antonucci, Adnan Darwiche, and Marco Zaffalon. “Tractable Bounding of Counterfactual Queries by Knowledge Compilation.” *Sixth Workshop on Tractable Probabilistic Modeling at UAI*, 2023.
- [HCD23] Yunqiu Han, Yizuo Chen, and Adnan Darwiche. “On the Complexity of Counterfactual Reasoning.” In *IJCAI*, pp. 5676–5684. ijcai.org, 2023.
- [HD24] Haiying Huang and Adnan Darwiche. “Causal Unit Selection using Tractable Arithmetic Circuits.” In *FLAIRS*. AAAI Press, 2024.
- [HM08] Terry A. Halpin and Tony Morgan. *Information modeling and relational databases (2. ed.)*. Morgan Kaufmann, 2008.
- [HR20] Miguel A. Hernán and James M. Robins. *Causal Inference: What If*. Boca Raton: Chapman & Hall/CRC, 2020.
- [HV06] Yimin Huang and Marco Valtorta. “Identifiability in Causal Bayesian Networks: A Sound and Complete Algorithm.” In *AAAI*, pp. 1149–1154. AAAI Press, 2006.
- [IC02] Jaime Shinsuke Ide and Fábio Gagliardi Cozman. “Random Generation of Bayesian Networks.” In *SBIA*, volume 2507 of *Lecture Notes in Computer Science*, pp. 366–375. Springer, 2002.
- [IR15] Guido W. Imbens and Donald B. Rubin. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press, 2015.
- [KC15] Johan Kwisthout and Cassio P. de Campos. “Lecture notes: Computational Complexity of Bayesian Networks.”, 2015.
- [LS88] Steffen L. Lauritzen and David 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.

- [MJE22] Ehsan Mokhtarian, Fateme Jamshidi, Jalal Etesami, and Negar Kiyavash. “Causal Effect Identification with Context-specific Independence Relations of Control Variables.” In *AISTATS*, volume 151 of *Proceedings of Machine Learning Research*, pp. 11237–11246. PMLR, 2022.
- [MLP22] Scott Mueller, Ang Li, and Judea Pearl. “Causes of Effects: Learning Individual Responses from Population Data.” In *IJCAI*, pp. 2712–2718. ijcai.org, 2022.
- [Pea86] Judea Pearl. “Fusion, Propagation, and Structuring in Belief Networks.” *Artif. Intell.*, **29**(3):241–288, 1986.
- [Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- [Pea93] Judea Pearl. “[Bayesian Analysis in Expert Systems]: Comment: Graphical Models, Causality and Intervention.” *Statistical Science*, **8**(3):266–269, 1993.
- [Pea95] Judea Pearl. “Causal Diagrams for Empirical Research.” *Biometrika*, **82**(4):669–688, 1995.
- [Pea99] Judea Pearl. “Probabilities Of Causation: Three Counterfactual Interpretations And Their Identification.” *Synth.*, **121**(1-2):93–149, 1999.
- [pea09] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, second edition, 2009.
- [pea18] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, 2018.
- [PJS17] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, 2017.
- [PNK15] Johan Pensar, Henrik J. Nyman, Timo Koski, and Jukka Corander. “Labeled directed acyclic graphs: a generalization of context-specific independence in directed graphical models.” *Data Min. Knowl. Discov.*, **29**(2):503–533, 2015.
- [RGW18] Denis Rosset, Nicolas Gisin, and Elie Wolfe. “Universal Bound on the Cardinality of Local Hidden Variables in Networks.” *Quantum Info. Comput.*, **18**(11–12):910–926, sep 2018.
- [RS86] Neil Robertson and Paul D. Seymour. “Graph Minors. II. Algorithmic Aspects of Tree-Width.” *J. Algorithms*, **7**(3):309–322, 1986.
- [SCD20] Yujia Shen, Arthur Choi, and Adnan Darwiche. “A New Perspective on Learning Context-Specific Independence.” In *PGM*, volume 138 of *Proceedings of Machine Learning Research*, pp. 425–436. PMLR, 2020.

- [SGS00] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search, Second Edition*. Adaptive computation and machine learning. MIT Press, 2000.
- [SP06] Ilya Shpitser and Judea Pearl. “Identification of Joint Interventional Distributions in Recursive Semi-Markovian Causal Models.” In *AAAI*, pp. 1219–1226. AAAI Press, 2006.
- [SP08] Ilya Shpitser and Judea Pearl. “Complete Identification Methods for the Causal Hierarchy.” *J. Mach. Learn. Res.*, **9**:1941–1979, 2008.
- [THK19] Santtu Tikka, Antti Hyttinen, and Juha Karvanen. “Identifying Causal Effects via Context-specific Independence Relations.” In *NeurIPS*, pp. 2800–2810, 2019.
- [TK17a] Santtu Tikka and Juha Karvanen. “Enhancing Identification of Causal Effects by Pruning.” *J. Mach. Learn. Res.*, **18**:194:1–194:23, 2017.
- [TK17b] Santtu Tikka and Juha Karvanen. “Simplifying Probabilistic Expressions in Causal Inference.” *J. Mach. Learn. Res.*, **18**:36:1–36:30, 2017.
- [TP00] Jin Tian and Judea Pearl. “Probabilities of causation: Bounds and identification.” *Ann. Math. Artif. Intell.*, **28**(1-4):287–313, 2000.
- [TP02a] Jin Tian and Judea Pearl. “A General Identification Condition for Causal Effects.” In *AAAI/IAAI*, pp. 567–573. AAAI Press / The MIT Press, 2002.
- [TP02b] Jin Tian and Judea Pearl. “On the Testable Implications of Causal Models with Hidden Variables.” In *UAI*, pp. 519–527. Morgan Kaufmann, 2002.
- [TP03] Jin Tian and Judea Pearl. “On the Identification of Causal Effects.” *Technical Report*, **R-290-L**, 2003.
- [Ver93] Thomas S. Verma. “Graphical Aspects of Causal Models.” *Technical Report*, **R-191**, 1993.
- [VP88] Thomas Verma and Judea Pearl. “Causal networks: semantics and expressiveness.” In *UAI*, pp. 69–78. North-Holland, 1988.
- [ZAC21] Marco Zaffalon, Alessandro Antonucci, and Rafael Cabañas. “Causal Expectation-Maximisation.” In *WHY Workshop, NeurIPS*, 2021. <https://arxiv.org/abs/2011.02912>.
- [ZLT19] Benito van der Zander, Maciej Liskiewicz, and Johannes Textor. “Separators and adjustment sets in causal graphs: Complete criteria and an algorithmic framework.” *Artif. Intell.*, **270**:1–40, 2019.

- [ZP96] Nevin L. Zhang and David L. Poole. “Exploiting Causal Independence in Bayesian Network Inference.” *J. Artif. Intell. Res.*, **5**:301–328, 1996.
- [ZTB22] Junzhe Zhang, Jin Tian, and Elias Bareinboim. “Partial Counterfactual Identification from Observational and Experimental Data.” In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 26548–26558. PMLR, 2022.