Lifted Causal Inference in Relational Domains

Author names withheld

Editor: Under Review for CLeaR 2024

Abstract
Lifted inference exploits symmetries in probabilistic graphical models by using a representative for indistinguishable objects, thereby speeding up query answering while maintaining exact answers. Even though lifting is a well-established technique for the task of probabilistic inference in relational domains, it has not yet been applied to the task of causal inference. In this paper, we show how lifting can be applied to efficiently compute causal effects in relational domains. More specifically, we introduce parametric causal factor graphs as an extension of parametric factor graphs incorporating causal knowledge and give a formal semantics of interventions therein. We further present the lifted causal inference algorithm to compute causal effects on a lifted level, thereby drastically speeding up causal inference compared to propositional inference, e.g., in causal Bayesian networks. In our empirical evaluation, we demonstrate the effectiveness of our approach.

Keywords: Causal graphical models, lifted probabilistic inference, interventional distributions

1. Introduction
A fundamental problem in the research field of artificial intelligence for an intelligent agent is to plan and act rationally in an environment following a relational structure. To compute the best possible action in a perceived state, the agent considers the available actions and chooses the one with the maximum expected utility. When computing the expected utility of an action that intervenes on a specific variable, it is crucial to deploy the semantics of an intervention instead of a typical conditioning on the observed (Pearl, 2009, Chapter 4). When calculating the effect of an intervention, a specific variable is set to a fixed value and all incoming probabilistic causal influences of this variable must be ignored for the specific query. It is therefore fundamental to deploy the semantics of an intervention instead of the typical conditioning to correctly determine the effect of an action.

Over the last years, causal graphical models became a widely used formalism to answer questions concerning the causal impact of a treatment variable on an outcome variable. These models combine probabilistic modeling with causal knowledge, enabling the computation of the effect of an action that intervenes on a particular variable. As our world is inherently relational (i.e., it consists of objects and relations between those objects), it is particularly important to have models that represent the relational structure between objects in addition to capturing causal knowledge. However, commonly applied causal graphical models focus on propositional representations while at the same time relational models lack the ability to efficiently apply causal knowledge for inference. Therefore, we aim to combine the best of both worlds to allow for efficient causal inference in relational domains. In particular, this paper deals with the problem of efficiently computing causal effects in models representing objects and their causal relationships to each other.

© 2024 .
Previous work. To perform causal effect estimation in causal graphical models, there has been a considerable amount of work and most of this work focuses on models of propositional data (Spirtes et al., 2000; Pearl, 2009). Some works extend propositional factor graphs (FGs) by adding edge directions to enable the computation of the effect of interventions (Frey, 2003; Winn, 2012). Maier et al. (2010) show that propositional models are insufficient to represent causal relationships within relational domains as required by real-world applications. To express causal dependencies within relational domains, Maier et al. (2013) introduce so-called relational causal models but focus on learning relational causal models from observed data. Most of the other related work covering relational causal models also deals with the problem of learning a causal model from relational data (e.g., Lee and Honavar, 2015, 2016). Prior work on the estimation of causal effects in relational domains applies propositional probabilistic inference (Arbour et al., 2016; Salimi et al., 2020) and thus does not scale for large graphs. Consequently, there is a lack of efficient algorithms to compute causal effects in relational domains. In probabilistic inference, lifting exploits symmetries in a relational model, allowing to carry out query answering more efficiently while maintaining exact answers (Niepert and Van den Broeck, 2014). First introduced by Poole (2003), parametric factor graphs (PFGs) and lifted variable elimination (LVE) allow to perform lifted probabilistic inference, i.e., to exploit symmetries in a probabilistic graphical model, resulting in significant speed-ups for probabilistic query answering in relational domains. Over time, LVE has been refined by many researchers to reach its current form (De Salvo Braz et al., 2005, 2006; Milch et al., 2008; Kisyński and Poole, 2009; Taghipour et al., 2013a; Braun and Möller, 2018). PFGs are well-studied for many years and have been developed further to efficiently perform probabilistic inference not only for single queries but also for sets of queries (Braun and Möller, 2016), to incorporate probabilistic inference over time (Gehrke et al., 2018, 2020), and, among other extensions, to allow for decision making by following the maximum expected utility principle (Gehrke et al., 2019a,b; Braun and Gehrke, 2022). Markov logic networks are another lifted representation and have been extended to incorporate maximum expected utility as well (Apsel and Brafman, 2012). Nevertheless, when a decision-making agent plans for the best action to take, previous works improperly apply conditioning, as also suggested by Russell and Norvig (2020, Chapter 16), instead of the notion of an intervention (i.e., actions are treated as evidence). Treating actions as evidence, however, is incorrect as noted by Pearl (2009, Chapter 4). To correctly handle the semantics of an action, the notion of an intervention (Pearl et al., 2016) has to be applied. Therefore, in this paper, we close the gap between PFGs and causal inference in relational domains by introducing parametric causal factor graphs as an extension of parametric factor graphs incorporating causal knowledge to allow for lifted causal inference, thereby enabling efficient decision making using the notion of an intervention.

Our contributions. PFGs are well-established models coming with LVE as a mature lifted inference algorithm, allowing for tractable probabilistic inference with respect to domain sizes in relational domains. We extend PFGs by incorporating causal knowledge, resulting in parametric causal factor graphs (PCFGs) for which we define a formal semantics of interventions. Having defined a formal semantics of interventions in PCFGs, we show how causal effects can be efficiently computed, even for multiple simultaneous interventions. More specifically, we introduce PCFGs as an extension of PFGs as well as the lifted causal inference (LCI) algorithm that operates on a lifted level to drastically speed up causal inference compared to propositional causal inference (e.g., in causal Bayesian networks). Apart from the theoretical investigation of PCFGs and the LCI algorithm, we provide an empirical evaluation confirming the efficiency of LCI.
Structure of this paper. Section 2 introduces both FGs and PFGs as undirected probabilistic graphical models. Thereafter, in Section 3, we define PCFGs as an extension of PFGs incorporating causal knowledge and provide a formal semantics of interventions in PCFGs. In Section 4, we introduce the LCI algorithm operating on a PCFG and show how LCI computes causal effects on a lifted level to avoid grounding the PCFG as much as possible. Afterwards, in our empirical evaluation in Section 5, we investigate the speed-up of LCI compared to performing propositional causal inference both in causal Bayesian networks and in directed FGs before we conclude in Section 6.

2. Preliminaries

We begin by introducing FGs as propositional probabilistic models and afterwards continue to define PFGs which combine probabilistic models and first-order logic to allow for tractable probabilistic inference with respect to domain sizes in relational domains. An FG is an undirected graphical model to compactly encode a full joint probability distribution between random variables (randvars) (Frey et al., 1997; Kschischang et al., 2001). Similar to a Bayesian network (Pearl, 1988), an FG factorises a full joint probability distribution into a product of factors.

Definition 1 (Factor Graph) An FG $G = (V, E)$ is a bipartite graph with node set $V = R \cup \Phi$ where $R = \{R_1, \ldots, R_n\}$ is a set of variable nodes (randvars) and $\Phi = \{\phi_1, \ldots, \phi_m\}$ is a set of factor nodes (functions). There is an edge between a variable node $R_i$ and a factor node $\phi_j$ in $E \subseteq R \times \Phi$ if $R_i$ appears in the argument list of $\phi_j$. A factor is a function that maps its arguments to a positive real number, called potential. The semantics of $G$ is given by

$$P_G = \frac{1}{Z} \prod_{j=1}^{m} \phi_j(A_j)$$

with $Z$ being the normalisation constant and $A_j$ denoting the randvars connected to $\phi_j$.

Example 1 Figure 1 shows a toy example of an FG modelling the relationships between a company’s revenue, its employee’s competences, and training of its employees. More specifically, there are randvars $\text{Qual}.t_i$ indicating the quality of a training program $t_i$, randvars $\text{Comp}.e_j$ describing the competence of an employee $e_j$, randvars $\text{Train}.e_j.t_i$ specifying whether employee $e_j$ has been trained with training program $t_i$, and a randvar $\text{Rev}$ denoting the revenue of the company. In this particular example, there is a single company with four employees alice, bob, dave, and eve and there are two training programs $t_1$ and $t_2$ each employee can be trained with. The randvars $\text{Qual}.t_i$, $\text{Comp}.e_j$, and $\text{Rev}$ can take one of the values $\{\text{low, medium, high}\}$ and the randvars $\text{Train}.e_j.t_i$ are Boolean. The factors $\phi_1$ encode the prior probability distribution for the quality of a training program, the $\phi_2$ encode the relationship between the quality of a training program and an employee being trained with that program, the $\phi_3$ encode the relationship between an employee being trained by a specific training program and the competence of that employee, and the $\phi_4$ encode the relationship between the competence of an employee and the revenue of the company. The input-output pairs of the factors are omitted for brevity.

We continue to define PFGs, first introduced by Poole (2003), which combine probabilistic models and first-order logic. In particular, PFGs use logical variables (logvars) as parameters in randvars to represent sets of indistinguishable randvars. Each set of indistinguishable randvars is represented by a so-called parameterised randvar (PRV), defined as follows.
Figure 1: A toy example of an FG modelling the interplay of a company’s revenue and its employee’s competences, which, in turn, can be improved by training employees with a specific training program. We omit the input-output pairs of the factors for brevity.

Definition 2 (Parameterised Random Variable) Let $R$ be a set of randvar names, $L$ a set of logvar names, and $D$ a set of constants. All sets are finite. Each logvar $L$ has a domain $D(L) \subseteq D$. A constraint is a tuple $(X, C_X)$ of a sequence of logvars $X = (X_1, \ldots, X_n)$ and a set $C_X \subseteq \times_{i=1}^n D(X_i)$. The symbol $\top$ for $C_X$ marks that no restrictions apply, i.e., $C_X = \times_{i=1}^n D(X_i)$. A PRV $R(L_1, \ldots, L_n)$, $n \geq 0$, is a syntactical construct of a randvar $R \in R$ possibly combined with logvars $L_1, \ldots, L_n \in L$ to represent a set of randvars. If $n = 0$, the PRV is parameterless and forms a propositional randvar. A PRV $A$ (or logvar $L$) under constraint $C$ is given by $A|_C = (L|_C)$, respectively. We may omit $|_\top$ in $A|_\top$ or $L|_\top$. The term $R(A)$ denotes the possible values (range) of a PRV $A$. An event $A = a$ denotes the occurrence of PRV $A$ with range value $a \in R(A)$.

Example 2 Consider $R = \{\text{Qual}, \text{Train}, \text{Comp}, \text{Rev}\}$ for quality, training, competence, and revenue, respectively, $L = \{E, T\}$ with $D(E) = \{\text{alice, bob, dave, eve}\}$ (employees) and $D(T) = \{t_1, t_2\}$ (training programs), combined into PRVs $\text{Qual}(T)$, $\text{Train}(E, T)$, $\text{Comp}(E)$, and $\text{Rev}$.

A parametric factor (parfactor) describes a function, mapping argument values to positive real numbers (called potentials), of which at least one is non-zero.

Definition 3 (Parfactor) Let $\Phi$ denote a set of factor names. We denote a parfactor $g$ by $\phi(A)|_C$ with $A = (A_1, \ldots, A_n)$ being a sequence of PRVs, $\phi: \times_{i=1}^n R(A_i) \rightarrow \mathbb{R}^+$ being a function with name $\phi \in \Phi$ mapping argument values to a positive real number called potential, and $C$ being a constraint on the logvars of $A$. We may omit $|_\top$ in $\phi(A)|_\top$. The term $lv(Y)$ refers to the logvars in some element $Y$, a PRV, a parfactor, or sets thereof. The term $gr(Y|_C)$ denotes the set of all instances (groundings) of $Y$ with respect to constraint $C$. 
Example 3 Take a look at the parfactor $g_3 = \phi_3(\text{Train}(E,T), \text{Comp}(E))\big|_T$. Following Examples 1 and 2, $g_3$ specifies $2 \cdot 3 = 6$ input-output pairs $\phi_3(\text{true, low}) = \varphi_1$, $\phi_3(\text{true, medium}) = \varphi_2$, $\phi_3(\text{true, high}) = \varphi_3$, and so on with $\varphi_i \in \mathbb{R}^+$. Further, we have $lv(g_3) = \{E,T\}$ and $gr(g_3) = \{\phi_3(\text{Train}(alice,t_1), \text{Comp}(alice)), \ldots, \phi_3(\text{Train}(eve,t_2), \text{Comp}(eve))\}$. Thus, in this specific example, the parfactor $g_3$ is able to represent a set of eight factors.

A PFG is then build from a set of parfactors $\{g_1, \ldots, g_m\}$.

Definition 4 (Parametric Factor Graph) A PFG $G = (V,E)$ is a bipartite graph with node set $V = A \cup G$ where $A = \{A_1, \ldots, A_n\}$ is a set of PRVs and $G = \{g_1, \ldots, g_m\}$ is a set of parfactors. A PRV $A_i$ and a parfactor $g_j$ are connected via an edge in $G$ (i.e., $\{A_i, g_j\} \in E$) if $A_i$ appears in the argument list of $g_j$. The semantics of $G$ is given by grounding and building a full joint distribution. With $Z$ as the normalisation constant and $A_k$ denoting the randvars connected to $\phi_k$, $G$ represents the full joint distribution

$$P_G = \frac{1}{Z} \prod_{g_j \in G} \prod_{\phi_k \in gr(g_j)} \phi_k(A_k).$$

Example 4 Figure 2 depicts a PFG $G$ consisting of four parfactors $g_1 = \phi_1(\text{Qual}(T))$, $g_2 = \phi_2(\text{Qual}(T), \text{Train}(E,T))$, $g_3 = \phi_3(\text{Train}(E,T), \text{Comp}(E))$, and $g_4 = \phi_4(\text{Comp}(E), \text{Rev})$. Assuming that both the ranges of the PRVs and the domains of the logvars follow Examples 1 to 3, $G$ is a lifted representation entailing equivalent semantics as the FG shown in Fig. 1. Each parfactor $g_1, \ldots, g_4$ represents a group of factors $\phi_1, \ldots, \phi_4$, respectively, and each PRV Qual, Train, Comp, and Rev represents a group of randvars. Consequently, the size (i.e., the number of parfactors and PRVs) of the PFG is independent of domain sizes whereas in the propositional FG, each additional employee or training increases the size of the graph.

The underlying assumption here is that there are indistinguishable objects, in this specific example employees, which can be represented by a representative. In particular, the assumption is that the competence of every employee has the same influence on the company’s revenue, i.e., all factors $\phi_i$ encode the same mappings (and the same holds for the $\phi_1, \phi_2, \text{and} \phi_3$, meaning training programs are indistinguishable as well). In other words, it is relevant for the company how many employees are competent but it does not matter which exact employees are competent. Note that the definition of PFGs also includes FGs, as every FG is a PFG containing only parameterless randvars.

In the following, we extend PFGs to incorporate causal knowledge, represented by directed edges defining cause-effect relationships between PRVs.
3. Parametric Causal Factor Graphs

PFGs are well-established models for which lifted inference algorithms exist to allow for tractable probabilistic inference with respect to domain sizes. Even though Frey (2003) introduces directed FGs on a ground level, PFGs have not yet been extended to incorporate causal knowledge.

Therefore, we now introduce PCFGs as an extension of PFGs incorporating causal knowledge and give a formal semantics of interventions therein. A PCFG extends an PFG by incorporating causal knowledge in form of directed edges—that is, each edge between two PRV (via a parfactor) describes a cause-effect relationship. For example, an edge \( A_1 \rightarrow A_2 \) indicates that \( A_1 \) is a cause of \( A_2 \) and, consequently, \( A_2 \) is an effect of \( A_1 \). In particular, in a PCFG, each parfactor is connected to a single child and zero or more parents, matching the definition of directed FGs in the ground case given by Frey (2003). Further, as commonly required in directed graphical models such as causal Bayesian networks, we restrict a PCFG to be acyclic.

**Definition 5 (Parametric Causal Factor Graph)** A PCFG is a directed graph \( G = (V, E) \) with node set \( V = A \cup G \) where \( A = \{ A_1, \ldots, A_n \} \) is a set of PRVs and \( G = \{ g_1, \ldots, g_m \} \) is a set of directed parfactors. A directed parfactor \( g = \phi(A)|^C_{\forall A} \) with \( A = (A_1, \ldots, A_k) \) being a sequence of PRVs, \( \phi: \times_{i=1}^k R(A_i) \mapsto \mathbb{R}^+ \) being a function, and \( C \) being a constraint on the logvars of \( A \), maps its argument values to a positive real number (potential). Again, we may omit \( |\top \) in \( \phi(A)|^{C}_{\top} \). \( A_i \in A \) denotes the child of \( \phi(A)|^{A_i} \) whereas all \( A_j \in A \) with \( j \neq i \) are the parents of \( \phi(A)|^{A_i} \). For each directed parfactor \( g \), there are edges \( (g, A_i) \in E \) and \( (A_j, g) \in E \) (for all \( A_j \neq A_i \)). A PCFG is an acyclic graph, that is, there is no sequence of edges \( (g_1, A_1), (A_1, g_2), \ldots, (g_k, A_k), (A_k, g_1) \) in \( E \). The semantics of \( G \) is given by grounding and building a full joint distribution, identical to the semantics of a PFG, i.e., with \( Z \) as the normalisation constant, \( A_k \), denoting the randvars connected to \( \phi_k \), and \( A_k^\ell \in A_k \) specifying the child of \( \phi_k \), \( G \) represents

\[
P_G = \frac{1}{Z} \prod_{g_i \in G} \prod_{\phi_k \in gr(g_i)} \phi_k(A_k)^{\rightarrow \ell}.
\]

**Example 5** Consider the PCFG \( G \) depicted in Fig. 3. \( G \) represents the same full joint probability distribution as the PFG shown in Fig. 2. In particular, both models are identical except for the fact that \( G \) contains directed edges instead of undirected edges between parfactors and PRVs. Each parfactor represents a group of directed factors and thus, grounding \( G \) results in a directed FG. Following previous examples by assuming \( D(T) = \{ t_1, t_2 \} \), for example \( g_1 = \phi_1(Qual(T))^{\rightarrow Qual(T)} \) represents \( gr(g_1) = \{ \phi_1(Qual(t_1))^{\rightarrow Qual(t_1)}, \phi_1(Qual(t_2))^{\rightarrow Qual(t_2)} \} \).

In the following, we denote the parents of a PRV \( A \) by \( Pa_G(A) = \{ \phi \rightarrow A_i \mid A = A_i \} \) and the child of a parfactor \( \phi \) by \( Ch_G(\phi(A) \rightarrow A_i) = A_i \) in a PCFG \( G \). If the context is clear, we omit the subscript \( G \). Before we define the semantics of an intervention in a PCFG, we briefly revisit the notion of \( d \)-separation in directed acyclic graphs and afterwards apply it to PCFGs.

3.1. \( d \)-Separation in Parametric Causal Factor Graphs

The notion of \( d \)-separation (Pearl, 1986) provides a graphical criterion to test for conditional independence in directed acyclic graphs. Frey (2003) translates the notion of \( d \)-separation to directed FGs. We build on the definition of \( d \)-separation in directed FGs to define \( d \)-separation in PCFGs.
Figure 3: An illustration of a PCFG encoding the same full joint probability distribution as the PFG given in Fig. 2. The only difference between the PCFG and the PFG is that the PCFG contains directed edges instead of undirected edges between PRVs and parfactors.

Definition 6 \((d\text{-}separation)\) Let \(G = (A \cup G, E)\) be a PCFG. Given three disjoint sets of randvars \(X, Y,\) and \(Z\) (subsets of \(\bigcup_{A \in A} \text{gr}(A)\)), we say that \(X\) and \(Y\) are conditionally independent given \(Z\), written as \(X \perp \perp Y | Z\), if the nodes in \(Z\) block all paths from the nodes in \(X\) to the nodes in \(Y\) in the directed FG obtained by grounding \(G\). A path is a connected sequence of edges \((A_i, g_i)\), \(\ldots, (A_\ell, g_\ell)\) with \((A_i, g_i) \in E\) or \((g_i, A_i) \in E\), i.e., a path is not restricted to follow the arrow directions of the edges. Note that it is therefore also possible for a path to pass from a parent of a factor to another parent of the factor. A path is blocked by the nodes in \(Z\) if

1. the path contains a variable from \(Z\), or
2. the path passes from a parent of a directed factor \(\phi\) to another parent of \(\phi\), and neither the child of \(\phi\) nor any of its descendants are in \(Z\).

The semantics of \(d\text{-}separation\) in PCFGs is defined on a ground level. However, it is possible to check for \(d\text{-}separation\) on a lifted level without having to ground the PCFG.

Example 6 Consider again the PCFG depicted in Fig. 3 and assume we want to check whether \(\text{Qual}(t_1) \perp \perp \text{Comp}(bob) | \text{Train}(bob, t_1)\) holds. In this case, we have assigned \(T = t_1\) and \(E = bob\), so we only need to examine paths involving this particular assignment of logvars. More specifically, all PRVs on the path with overlapping logvars, i.e., all PRVs having \(T\) or \(E\) as a logvar, are bound to the same assignment. Therefore, all paths from \(\text{Qual}(t_1)\) to \(\text{Comp}(bob)\) pass through \(\text{Train}(bob, t_1)\), meaning the conditional independence statement in question holds. Note that if there were other paths involving PRVs with non-overlapping logvars, i.e., logvars not involved in the sets \(X\) and \(Y\), all randvars represented by those PRVs need to be in \(Z\) to block those paths.

The concept of \(d\text{-}separation\) is important for the computation of the effect of an intervention in the sense that all non-causal paths, so-called backdoor paths, need to be blocked. We next show how these backdoor paths are blocked when performing an intervention and give a formal semantics of an intervention in a PCFG.

3.2. Semantics of Interventions in Parametric Causal Factor Graphs

To correctly handle the semantics of an action, for example in the setting of a decision-making agent planning for the best action to take, we have to differentiate between seeing (conditioning) and doing (intervention). Let us take a look at the PCFG shown in Fig. 3 again. If we observe (see) an event \(\text{Train}(bob, t_1) = true\), our belief about the probability distribution of \(\text{Qual}(t_1)\) might change. More specifically, the probability of \(t_1\) having a high quality might be higher when observing \(\text{Train}(bob, t_1) = true\) than without the observation under the assumption that the probability of
training an employee increases if the quality of a training program is high. However, if we are interested
in the effect an action setting \( \text{Train}(\text{bob}, t_1) \) to \textit{true}, denoted as \( \text{do}(\text{Train}(\text{bob}, t_1)) = \text{true} \), has
on the remaining PRVs, we have to ensure that the belief about the probability of \( t_1 \) having a
high quality remains unchanged as the action itself has no influence on the probability distribution
of \( \text{Qual}(t_1) \). Therefore, it is crucial to avoid the propagation of information against the edge
directions whenever we are interested in the effect of an action. That is, if we are interested in the effect
a specific randvar \( R' \) has on another randvar \( R \), all so-called backdoor paths from \( R \) to \( R' \) must be
blocked. A backdoor path is a non-causal path, i.e., a backdoor path from \( R \) to \( R' \) is a path that
remains after removing all outgoing edges of \( R \).

To account for backdoor paths and correctly handle the semantics of an action, we employ the
notion of an intervention. An intervention on a randvar \( R \), denoted as \( \text{do}(R = r) \) with \( r \in \mathcal{R}(R) \),
changes the structure of a PCFG by removing all parent edges of \( R \) and setting \( R \) to the value \( r \).
By removing the parent edges, all backdoor paths are removed. Formally, the semantics of an
intervention in a PCFG is defined as below, following the definition of an intervention in Bayesian
networks provided by Pearl et al. (2016).

\textbf{Definition 7 (Intervention)} Let \( R = \{R_1, \ldots, R_n\} \) be the set of randvars obtained by grounding
a PCFG \( G = ( \mathcal{A} \cup \mathcal{G}, \mathcal{E}) \), i.e., \( R = \bigcup_{A \in \mathcal{A}} \text{gr}(A) \). An intervention \( \text{do}(R_i = r_i, \ldots, R_k = r_k) \)
changes the underlying probability distribution such that each factor \( \phi(R'_1, \ldots, R'_1, \ldots, R'_\ell) \rightarrow R_i \)
with \( R'_i \in \{R_1, \ldots, R_n\} \) is replaced by a factor \( \phi'(R'_1, \ldots, R'_1, \ldots, R'_\ell) \rightarrow R'_i \) with
\[
\phi'(R'_1 = r'_1, \ldots, R'_i = r'_i, \ldots, R'_\ell = r'_\ell) \rightarrow R'_i = \begin{cases} 1 & \text{if } r_i = r'_i \\ 0 & \text{if } r_i \neq r'_i. \end{cases}
\]
The remaining \( \phi(R'_1, \ldots, R'_1, \ldots, R'_\ell) \rightarrow R'_i \) with \( R'_i \notin \{R_1, \ldots, R_n\} \) remain unchanged.

By fixing the values of all parent factors, all parent influences are (virtually) removed from the
model and hence initial backdoor paths are (virtually) removed from the model as well. Having
defined the semantics of an intervention in a PCFG, we are now interested in efficiently computing
interventional distributions, i.e., the result of queries that contain \textit{do}-expressions. Note that in a
PCFG, all causal effects are identifiable per definition and thus, we do not have to rewrite a query
containing \textit{do}-expressions according to the \textit{do}-calculus (Pearl, 1995) to obtain an equivalent query
free of \textit{do}-expressions. In particular, we do not estimate causal effects from observed data but
instead compute them in a fully specified model as every PCFG encodes a full joint probability
distribution which we can modify according to the definition of an intervention and afterwards
query the modified distribution to answer any query containing \textit{do}-expressions.

We next introduce the LCI algorithm, which handles interventions in PCFGs efficiently by
directly applying the semantics of an intervention on a lifted level.

\textbf{4. Efficient Causal Effect Computation in Parametric Causal Factor Graphs}

Now that we have introduced PCFGs, we study the problem of efficiently computing the effect of
interventions in PCFGs. A major advantage of using PCFGs instead of propositional models such as
causal Bayesian networks is that we mostly do not have to fully ground the model to compute the ef-
fect of interventions. Consider again the PCFG illustrated in Fig. 3 and assume we want to compute
the interventional distribution \( P(\text{Rev} \mid \text{do}(\text{Train}(\text{bob}, t_1))) = \text{true} \) in \( G \). Note that when intervening on a randvar, we have to treat it differently than other randvars in the same group on which we do not intervene. An intervention \( \text{do}(\text{Treat}(\text{bob}, t_1)) = \text{true} \) sets the value of \( \text{Treat}(\text{bob}, t_1) \) to \text{true} and thus, we have to treat \text{bob} different from \text{alice}, \text{dave}, and \text{eve}—in other words, not all employees are indistinguishable anymore. Nevertheless, and this is the crucial point, we can still treat \text{alice}, \text{dave}, and \text{eve} as indistinguishable when computing the interventional distribution.

### 4.1. The Lifted Causal Inference Algorithm

We now introduce the LCI algorithm to compute the interventional distribution \( P(R_1, \ldots, R_\ell \mid \text{do}(R'_1 = r'_1, \ldots, R'_k = r'_k)) \) in a PCFG \( G \). The entire LCI algorithm is shown in Alg. 1.

First, LCI splits the parfactors in \( G \) based on the intervention variables \( R'_i \in \{R'_1, \ldots, R'_k\} \). In particular, splitting parfactors in \( G \) results in a modified PCFG \( G' \) entailing equivalent semantics as \( G \) (De Salvo Braz et al., 2005). The procedure of splitting a parfactor works as follows. Recall that \( R'_i = A(L_1 = l_1, \ldots, L_j = l_j) \), \( l_j \in \mathcal{D}(L_1), \ldots, l_j \in \mathcal{D}(L_j) \), is a particular instance of a \( \text{PRV} A(L_1, \ldots, L_j) \), that is, it holds that \( R'_i \in \mathcal{gr}(A) \). The idea behind the splitting procedure is that we would like to separate \( \mathcal{gr}(A) \) into two sets \( \mathcal{gr}(A) \setminus \{R'_i\} \) and \( \{R'_i\} \), as \( R'_i \) has to be treated differently than the remaining instances of \( A \). Therefore, every parfactor \( g \) for which there is an instance \( \phi \in \mathcal{gr}(g) \) such that \( R'_i \) appears in the argument list of \( \phi \) is split. Formally, splitting a parfactor \( g \) replaces \( g \) by two parfactors \( g'_c \) and \( g''_c \) and adapts the constraints of \( g'_c \) and \( g''_c \). The constraints \( C' \) and \( C'' \) are altered such that the inputs of \( g'_c \) are restricted to all sequences that contain \( R'_i \) and the inputs of \( g''_c \) are restricted to the remaining input sequences. After the splitting procedure, the semantics of the model remains unchanged as the groundings of \( G' \) are still the same as the groundings of the initial model \( G \)—they are just arranged differently across the sets of ground instances. Having completed the split of all respective parfactors, LCI next modifies the parents of \( R'_i \), i.e., the underlying probability distribution encoded by \( G' \) is modified according to the semantics of the intervention \( \text{do}(R'_i = r'_i) \) with \( r'_i \in \mathcal{R}(R'_i) \). More specifically, as \( R'_i \) is
Comp

Figure 4: A visualisation of the modified PCFG obtained after altering the PCFG shown in Fig. 3 by splitting \( g_2 \) and \( g_3 \) to separate \( \text{Train}(bob, t_1) \) from \( \text{Train}(E, T) \). Here, the constraints \( C_2 \) as well as \( C_3 \) include all instances of \( \text{Train}(E, T) \) except for \( \text{Train}(bob, t_1) \) and \( C'_2 \) as well as \( C'_3 \) are restricted to the single instance \( \text{Train}(bob, t_1) \) of \( \text{Train}(E, T) \). Note that the graph size remains significantly smaller than for the fully grounded model.

fixed on \( r'_i \), all parents \( \phi \in \text{Pa}_{G'}(R'_i) \) of \( R'_i \) are altered such that all input sequences assigning \( R'_i = r'_i \) map to the potential value one while all other input sequences map to zero. Finally, LCI computes the result for \( P(R_1, \ldots, R_\ell | \text{do}(R'_1 = r'_1, \ldots, R'_k = r'_k)) \) in the original model \( G \). To perform query answering in \( G' \), LVE can be applied to \( G' \) by simply ignoring the edge directions as the semantics of a PFG and a PCFG are defined identically.

Before we continue to examine the correctness of Alg. 1, we take a look at an example.

**Example 7** Consider again the PCFG \( G \) shown in Fig. 3 and assume we would like to compute \( P(\text{Rev} | \text{do} (\text{Train}(bob, t_1)) = \text{true}) \). As \( \text{Train}(bob, t_1) \) is a particular instance of \( \text{Train}(E, T) \), we have to split the parfactors \( g_2 \) and \( g_3 \) while \( g_1 \) as well as \( g_4 \) keep \( \top \) as their constraint. Figure 4 shows the modified PCFG \( G' \) obtained after splitting \( g_2 \) and \( g_3 \) based on the intervention on \( \text{Train}(bob, t_1) \). In \( G' \), \( \text{Train}(bob, t_1) \) is now a separate node in the graph, connected to two newly introduced parfactors. In particular, \( g_2 \) has been replaced by two parfactors \( g'_2_{|C_2} \) and \( g'_{2|C_2'} \) with constraints \( C_2 = (((E, T), \{(alice, t_1), (alice, t_2), (bob, t_1), (dave, t_1), (dave, t_2), (eve, t_1), (eve, t_2)) \}) \) and \( C'_2 = (((E, T), \{(bob, t_1)) \}) \). In other words, \( g'_2_{|C_2} \) is restricted to all instances of \( \text{Train}(E, T) \) except for \( \text{Train}(bob, t_1) \) and \( g'_{2|C_2'} \) is restricted to the instance \( \text{Train}(bob, t_1) \) of \( \text{Train}(E, T) \). Analogously, \( g_3 \) has been replaced by two parfactors \( g'_3_{|C_3} \) and \( g'_{3|C_3'} \). To incorporate the semantics of \( \text{do}(\text{Train}(bob, t_1)) = \text{true} \), LCI next modifies the parents of \( \text{Train}(bob, t_1) \), i.e., LCI modifies \( g'_{2|C_2'} \) in this example. More specifically, \( g'_{2|C_2'}(\text{Qual}(t_1) = q, \text{Train}(bob, t_1) = \text{true}) \) is set to one and \( g'_{2|C_2'}(\text{Qual}(t_1) = q, \text{Train}(bob, t_1) = \text{false}) \) is set to zero for all \( q \in \mathcal{R}(\text{Qual}(t_1)) \). Finally, LVE can be run to compute \( P(\text{Rev} | \text{do}(\text{Train}(bob, t_1)) = \text{true}) \) in the original model \( G \).

Due to the splitting of parfactors, it might be the case that there are PRVs in \( G' \) having more parents than they previously had in the original model \( G \), as with \( \text{Comp}(E) \) in Fig. 4. The semantics of the model, however, remains unchanged because \( \bigcup_{g \in G} \text{gr}(g) = \bigcup_{g \in G'} \text{gr}(g) \). Given the way we specified the semantics of an intervention in a PCFG, we can show that LCI correctly computes the effect of interventions. In particular, as LCI directly applies Def. 7 by setting the parent factors of
all variables we intervene on accordingly, the semantics of the modified model $G'$ is equivalent to
the semantics of interventions from Def. 7.

**Corollary 1** Algorithm 1 computes the interventional distribution according to Def. 7.

Moreover, directly applying Def. 7 allows LCI to exploit the established LVE algorithm. By deploy-
ing LVE, LCI is able to perform tractable inference (i.e. LCI runs in polynomial time) with respect
to domain sizes for all PCFGs belonging to the class of domain-liftable models (Taghipour et al.,
2013b). The class of domain-liftable models includes all PCFGs containing only parfactors with at
most two logvars and all PCFGs containing only PRVs having at most one logvar.

**Corollary 2** Algorithm 1 is able to perform tractable probabilistic inference with respect to domain
sizes for the class of domain-liftable models.

To summarise, LCI is a simple, yet effective algorithm to perform lifted causal inference, even for
interventions on large groups of randvars, as we investigate next.

### 4.2. Handling Interventions on Groups of Random Variables

LCI is able to handle both interventions on a single (ground) randvar as well as interventions on a
conjunction of multiple randvars efficiently. In particular, when intervening on multiple randvars
at the same time, LCI is able to treat those randvars as a group. For example, recall the employee
example and assume we want to train multiple employees simultaneously as a training program is
mostly offered not only for a single employee but for a group of employees. Then, it is not necessary
to split all trained employees into separate groups—it is sufficient to differentiate between trained
employees and all remaining employees. Formally, the interventions $do(R_1' = r_1', \ldots, R_k' = r_k')$
on an arbitrary set of randvars $\{R_1', \ldots, R_k'\}$ can thus efficiently be handled by splitting the parfactors
in $G$ such that all $R_i'$ that are represented by the same PRV $A$ and set to the same value $r_i'$ remain
grouped, equal to splitting on constraints in LVE. More specifically, LCI needs just a single split per
group and thus avoids manipulating the parents of each individual randvar separately. Furthermore,
it is also possible to intervene on a PRV (instead of intervening on a randvar). The semantics of
an intervention on a PRV $A$ is given by $do(A = a) = do(R_1 = a, \ldots, R_k = a)$ with $gr(A) =
\{R_1, \ldots, R_k\}$. Again, LCI is able to treat all randvars represented by $A$ as a group and therefore is
not required to split the group. In contrast, in a propositional model, every object has to be treated
individually and therefore the parents for each randvar need to be manipulated separately.

Next, we investigate the practical performance of PCFGs and, in particular, the LCI algorithm
for the computation of interventional distributions.

### 5. Experiments

In this section, we evaluate the run times needed to compute the result of interventional queries
in Bayesian networks, directed FGs, and PCFGs. For our experiments, we use a slightly modi-
fied version of the PCFG given in Fig. 3 which can directly be translated into a Bayesian network
without having to combine multiple parent factors into a single conditional probability table. More
specifically, to obtain the corresponding directed FG, we simply ground the PCFG and to obtain
the equivalent Bayesian network, we use the transformation from directed FG to Bayesian network
given by Frey (2003). Note that the PCFG used in our experiments to demonstrate the practical
efficiency of lifted causal inference is rather small with four parfactors and PRVs, respectively, and the gain we obtain from lifted inference further increases with models consisting of more PRVs.

We test the required run time for each of the three graphical models on different graph sizes by setting the domain size of the employees to \(d = 8, 16, 32, \ldots, 4096\) and having a single training program for each choice of \(d\) (i.e., \(|D(E)| = d\) and \(|D(T)| = 1\)). Figure 5 shows the run times needed to compute an interventional distribution for a single intervention in the modified graph when running variable elimination on the directed FG, variable elimination on the Bayesian network, and LVE on the PCFG. The results emphasise that the LCI algorithm, which internally exploits LVE, overcomes scalability issues for large domain sizes as the run time of LVE, in contrast to the run times of variable elimination on the Bayesian network and the directed FG, does not exponentially increase with \(d\) (y-axis is log-scaled). To conclude, PCFGs not only provide us with expressive probabilistic graphical models for relational domains but also enable us to drastically speed up causal inference by reasoning over sets of indistinguishable objects.

6. Conclusion

We introduce PCFGs to combine lifted probabilistic inference in relational domains with causal inference, thereby allowing for lifted causal inference. PCFGs provide a powerful formalism to represent causal relationships in relational domains that has been missing so far. To leverage the power of lifted inference for causal effect computation, we present the LCI algorithm which operates on a lifted level and thus allows us to perform tractable inference with respect to domain sizes in relational domains. LCI is a simple, yet effective algorithm to compute the effect of (multiple simultaneous) interventions, and builds on the well-founded LVE algorithm, thereby allowing LCI to be plugged into parameterised decision models (Gehrke et al., 2019b) to compute the maximum expected utility in accordance with Pearl (2009).

PCFGs open up interesting directions for future work. A basic problem is to learn a PCFG directly from a relational database. Following up on learning PCFGs from data, another constitutive problem for future research is to relax the assumption of having a fully directed PCFG at hand, i.e., to allow PCFGs to contain both directed and undirected edges at the same time and investigate the implications for answering causal queries.
References


