TECHNICAL REPORT
R-117
First version: Aug, 2024
Last version: Jul, 2025

Testing Causal Models with Hidden Variables in Polynomial Delay via Conditional Independencies

Hyunchai Jeong[*1], **Adiba Ejaz**[*2], **Jin Tian**[3], **Elias Bareinboim**[2]
[1]Purdue University
[2]Columbia University
[3]Mohamed bin Zayed University of Artificial Intelligence
jeong3@purdue.edu, adiba.ejaz@cs.columbia.edu, jin.tian@mbzuai.ac.ae, eb@cs.columbia.edu

Abstract

Testing a hypothesized causal model against observational data is a key prerequisite for many causal inference tasks. A natural approach is to test whether the conditional independence relations (CIs) assumed in the model hold in the data. While a model can assume exponentially many CIs (with respect to the number of variables), testing all of them is both impractical and unnecessary. Causal graphs, which encode these CIs in polynomial space, give rise to local Markov properties that enable model testing with a significantly smaller subset of CIs. Model testing based on local properties requires an algorithm to list the relevant CIs. However, existing algorithms for realistic settings with hidden variables and non-parametric distributions can take exponential time to produce even a single CI constraint. In this paper, we introduce the c-component local Markov property (C-LMP) for causal graphs with hidden variables. Since C-LMP can still invoke an exponential number of CIs, we develop a *polynomial delay* algorithm to list these CIs in poly-time intervals. To our knowledge, this is the first algorithm that enables poly-delay testing of CIs in causal graphs with hidden variables against arbitrary data distributions. Experiments on real-world and synthetic data demonstrate the practicality of our algorithm.

Code — https://github.com/CausalAILab/ListConditionalIndependencies

1 Introduction

Causal models are the daily bread of many fields of research (Pearl 2000; Spirtes, Glymour, and Scheines 2001), but tools for testing them are lacking. In various studies, researchers posit a causal model and use it to compute causal effects from data (Tennant et al. 2020; Hoover 1990; King et al. 2004; Sverchkov and Craven 2017; Robins, Hernan, and Brumback 2000; Rotmensch et al. 2017). The model imposes testable constraints on the statistics of the data collected. Before using the model for causal inference, it's crucial to test if these constraints are met, and adjust the model as needed (Pearl 1995, 2000; Bareinboim and Pearl 2016; Malinsky 2024; Ankan and Textor 2022).

Causal directed acyclic graphs (DAGs) are one popular model for causal assumptions (Pearl 2000; Spirtes, Glymour, and Scheines 2001). Conditional independencies (CIs) are the most basic constraint that a causal DAG imposes on observational data. The study of CIs in the context of graphical models dates back to at least the 1980's (Pearl 1988; Dawid 1979; Spirtes et al. 1998; Pearl 1998; Pearl and Meshkat 1999; Pearl 2000). A classic problem in this line of research is: *given observational data and a hypothesized causal graph, do all the CIs implied by this graph hold in the data?* If the answer is no, the DAG may be revised.

A key idea in the early literature of graphical models was to use a DAG to represent the constraints of probability distributions. A multivariate probability distribution may encode exponentially many CIs with respect to the number of variables. A DAG can encode these CIs in polynomial space. The *d-separation* criterion allows us to derive the CIs encoded in a DAG (Pearl 1988). The *global Markov property* of a DAG is the set of all CIs encoded in it (Pearl 1988). There is also a well-known *local Markov property* for DAGs (Pearl 1988; Lauritzen et al. 1990), which states that each variable must be conditionally independent of its non-descendants given its parents. Since the CI relation is a semi-graphoid, the linearly many CIs of the local Markov property together imply the exponentially many CIs of the global Markov property. This means that to test a DAG against observational data, it suffices to perform a linear number of CI tests as given by the local Markov property. For concreteness, consider the DAG $\mathcal{G}^1$ in Fig. 1a and assume all variables $\{A, B, \ldots, H, U_1, U_2, U_3\}$ are observed. Though $\mathcal{G}^1$ encodes 35787 CIs, only 11 need testing by the local Markov property. For example, if we test that $F \perp\!\!\!\perp \{A, B\} \mid \{C\}$, we do not need to test that $F \perp\!\!\!\perp \{A\} \mid \{B, C\}$, since the former implies the latter by the weak union axiom.

Unobserved confounding is a widespread phenomenon in real-world settings (Fisher 1936). It occurs when a hidden variable causally affects two or more observed variables. The local Markov property can be used to test *Markovian* causal DAGs, which represent models without unobserved confounding. However, it cannot be used to test *non-Markovian* DAGs, which represent models with unobserved confounding. This is because if the parents of a variable are partially unobserved, we cannot test CIs that require conditioning on these parents (Fig. 1a). Since the assumption of no unobserved confounding rarely holds in practice, alternative ways to test non-Markovian DAGs have been developed (Tian and

[*]These authors contributed equally.

Pearl 2002b; Kang and Tian 2009; Geiger and Meek 1998, 1999; Richardson 2003, 2009; Hu and Evans 2023). Despite their power, these works either (a) make strong assumptions on the DAG or probability distribution, or (b) do not provide an algorithm to query their required CI tests in poly-time intervals, with naive algorithms taking exponential time to output a single CI constraint.

Summary of contributions. We give the first efficient algorithm for testing causal DAGs with hidden variables via conditional independencies. This enables researchers to test their causal assumptions using observational data prior to inference. Importantly, our approach extends to arbitrary data distributions and networks of unobserved confounding.

This result builds on a newer, fine-grained characterization of CIs in graphs based on a new construct called ancestral c-components (i.e., connected components in the bidirected skeleton). In particular, we show that $O(n2^s)$ CI tests (Prop. 1) are required to test a DAG on n variables whose largest c-component has size s. This is an exponential improvement over naively testing all $\Theta(4^n)$ CI constraints encoded in the DAG. The upshot is largest for DAGs with many variables but small c-components. For instance, the DAG $\mathcal{G}^2$ in Fig. 1b implies 753 CIs, but only 5 really need testing. More specifically, our contributions are as follows:

1. We introduce the c-component local Markov property, or C-LMP (Def. 5). We show that C-LMP and the global Markov property are equivalent, admitting the same set of probability distributions for a given DAG. We then show an important property of C-LMP: a one-to-one mapping between the CI constraints it invokes and *ancestral c-components* (Thm. 2).
2. Building on this characterization, we develop the first algorithm (LISTCI) capable of listing all testable CI constraints of C-LMP in *polynomial delay* (Thm. 3). On a DAG with n nodes and m edges, LISTCI takes $O(n^2(n+m))$ time to return each new CI constraint, if one exists, or exit when it has exhausted all CI constraints.

Experiments with synthetic data and a real-world protein signaling dataset (Sachs et al. 2005) corroborate the theoretical findings. For the sake of space, proofs are provided in Appendix C.

2 Preliminaries

Notation. We use capital letters to denote variables (X), small letters for their values (x), and bold letters for sets of variables ($\mathbf{X}$) and their values ($\mathbf{x}$). The probability distribution over a set of variables $\mathbf{X}$ is denoted by $P(\mathbf{X})$. We consistently use $P(\mathbf{x})$ as abbreviations for probabilities $P(\mathbf{X}=\mathbf{x})$. For disjoint sets of variables $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$, we use $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$ to denote that $\mathbf{X}$ and $\mathbf{Y}$ are conditionally independent given $\mathbf{Z}$.

Structural causal models. The basic framework of our analysis rests on *structural causal models* (SCMs) (Pearl 2000, Def. 7.1.1). An SCM $\mathcal{M}$ is a quadruple $\mathcal{M} = \langle \mathbf{V}, \mathbf{U}, \mathcal{F}, P(\mathbf{u}) \rangle$ where $\mathbf{V}$ and $\mathbf{U}$ are sets of endogeneous and exogeneous variables, respectively. $\mathcal{F}$ is a set of functions: each $V \in \mathbf{V}$ is a function $f_V(\mathbf{PA_V}, \mathbf{U_V})$ of its endogeneous and exogeneous parents, $\mathbf{PA_V} \subseteq \mathbf{V}$ and $\mathbf{U_V} \subseteq \mathbf{U}$ respectively. $P(\mathbf{u})$ is a joint distribution over $\mathbf{U}$. Each SCM $\mathcal{M}$ induces an observed distribution $P(\mathbf{v})$ over $\mathbf{V}$. An SCM is said to be *Markovian* if $\mathbf{U}_V, \mathbf{U}_W$ are independent for every distinct $V, W \in \mathbf{V}$, and *non-Markovian* otherwise. For a more detailed survey on SCMs, we refer to (Pearl 2000; Bareinboim et al. 2022).

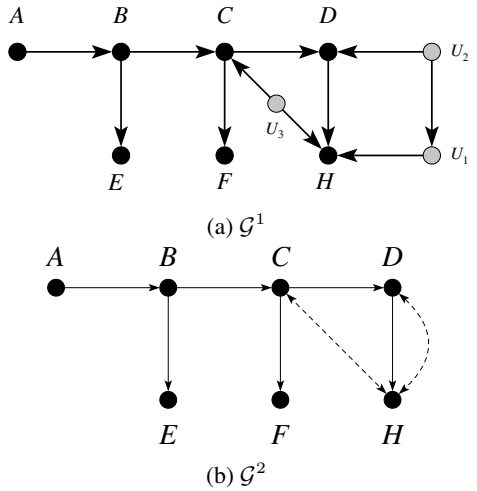


Figure 1: (a) A causal DAG $\mathcal{G}^1$ in which the local Markov property implies the CI: $H \perp\!\!\!\perp \{A, B, C, E, F\} \mid \{D, U_1, U_3\}$. If U_1 and U_3 are unobserved, we cannot test this CI. (b) We project $\mathcal{G}^1$ onto its observed variables to get $\mathcal{G}^2$. In $\mathcal{G}^2$, the c-component local Markov property invokes the testable CI: $H \perp\!\!\!\perp \{A, E, F\} \mid \{B, C, D\}$.

Causal graphs. The causal graph $\mathcal{G}$ for an SCM $\mathcal{M} = \langle \mathbf{V}, \mathbf{U}, \mathcal{F}, P(\mathbf{u}) \rangle$ is constructed as follows: (1) add a vertex for every $V \in \mathbf{V}$ (2) add an edge $V_i \to V_j$ for every $V_i, V_j \in \mathbf{V}$ if $V_i \in \mathbf{PA_{V_j}}$ (3) add a dashed bidirected edge between V_i, V_j if $\mathbf{U}_i, \mathbf{U}_j$ are correlated or $\mathbf{U}_i \cap \mathbf{U}_j \neq \emptyset$. $\mathcal{G}$ is said to be Markovian if it contains only directed edges, and semi-Markovian otherwise.

We denote the sets of parents, ancestors, and descendants of $\mathbf{X}$ (including $\mathbf{X}$ itself) in $\mathcal{G}$ as $Pa(\mathbf{X})$, $An(\mathbf{X})$, and $De(\mathbf{X})$, respectively. The set of non-descendants of $\mathbf{X}$ in $\mathcal{G}$ is denoted $Nd(\mathbf{X}) = \mathbf{V} \setminus De(\mathbf{X})$, which does not include $\mathbf{X}$ itself. The set of spouses of $\mathbf{X}$ in $\mathcal{G}$ is $Sp(\mathbf{X}) = \bigcup_{X \in \mathbf{X}} \{Y \mid Y \leftrightarrow X\}$. $\mathbf{X}$ is said to be an *ancestral set* if it contains its own ancestors, i.e., $\mathbf{X} = An(\mathbf{X})$. We use $\mathcal{G}_\mathbf{X}$ to denote the induced subgraph of $\mathcal{G}$ on $\mathbf{X} \subseteq \mathbf{V}$. A subscript $\mathcal{G}'$, e.g., $An(\mathbf{X})_{\mathcal{G}'}$ indicates that the set is computed from the subgraph $\mathcal{G}'$. We omit the subscript when clear from context. An ordering $\mathbf{V}^\prec$ on variables $\mathbf{V}$ is said to be consistent with $\mathcal{G}$ (i.e., a topological ordering) if for any $X, Y \in \mathbf{V}$, $X \prec Y$ implies $Y \notin An(X)_\mathcal{G}$. Let $\mathbf{V}^{\leq X} = \{Y \mid Y \prec X \text{ or } Y = X\}$.

Semi-Markovianity vs Non-Markovianity. A non-Markovian causal DAG $\mathcal{G}$ can be constructed for a non-Markovian SCM by making the exogenous variables $\mathbf{U}$ explicit. A non-Markovian DAG with arbitrary hidden variables can be 'projected' onto a semi-Markovian causal DAG $\mathcal{G}'$ which imposes exactly the same CI constraints over the observed variables (Tian and Pearl 2002b). In $\mathcal{G}'$,

each unobserved variable is (i) a parent of at most two observed variables and (ii) made implicit by adding a dashed bidirected edge between its two children. The complexity of the latent structure is irrelevant to the CIs over observed variables. Therefore, we work with semi-Markovian graphs for model testing.

d-separation. A node W on a path π is said to be a collider on π if W has converging arrows into W in π, e.g., $\rightarrow W \leftarrow$ or $\leftrightarrow W \leftarrow$. π is said to be blocked by a set $\mathbf{Z}$ if there exists a node W on π satisfying one of the following two conditions: 1) W is a collider, and neither W nor any of its descendants are in $\mathbf{Z}$, or 2) W is not a collider, and W is in $\mathbf{Z}$ (Pearl 1988). Given disjoint sets $\mathbf{X}, \mathbf{Y}$, and $\mathbf{Z}$ in $\mathcal{G}$, $\mathbf{Z}$ is said to *d-separate* $\mathbf{X}$ from $\mathbf{Y}$ in $\mathcal{G}$ if and only if $\mathbf{Z}$ blocks every path from a node in $\mathbf{X}$ to a node in $\mathbf{Y}$ according to the d-separation criterion (Pearl 1988). If $\mathbf{Z}$ d-separates $\mathbf{X}$ from $\mathbf{Y}$ in $\mathcal{G}$ (written $\mathbf{X} \perp_d \mathbf{Y} \mid \mathbf{Z}$), then $\mathbf{X}$ is conditionally independent of $\mathbf{Y}$ given $\mathbf{Z}$ in any observational distribution consistent with $\mathcal{G}$ (Pearl 1988; Richardson 2003).

Definition 1. (C-component) (Tian and Pearl 2002a) A set of variables $\mathbf{C} \subseteq \mathbf{V}$ in a causal graph $\mathcal{G}$ is said to be a confounded component (c-component, for short) if there is a path of only bidirected edges connecting any $V_i, V_j \in \mathbf{C}$, and $\mathbf{C}$ is maximal.

For a variable $X \in \mathbf{V}$, $\mathcal{C}(X)_{\mathcal{G}}$ denotes the c-component containing X in $\mathcal{G}$.

Previously, we have referred to the set of all CIs encoded in a DAG. We define this formally.

Definition 2. (Global Markov Property (GMP)) (Pearl 1988; Geiger, Verma, and Pearl 1989) A probability distribution $P(\mathbf{v})$ over a set of variables $\mathbf{V}$ is said to satisfy the global Markov property for a causal graph $\mathcal{G}$ if, for arbitrary disjoint sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subset \mathbf{V}$ with $\mathbf{X}, \mathbf{Y} \neq \emptyset$,

$$\mathbf{X} \perp_d \mathbf{Y} | \mathbf{Z} \implies \mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z} \text{ in } P(\mathbf{v}).$$

Various local Markov properties have been developed which identify a subset of the CIs invoked by GMP that imply all others. A prominent example is the local Markov property for Markovian DAGs.

Definition 3 (The Local Markov Property (LMP) (Pearl 1988; Lauritzen et al. 1990; Lauritzen 1996)[1])**.** A probability distribution $P(\mathbf{v})$ over a set of variables $\mathbf{V}$ is said to satisfy the local Markov property for a given Markovian DAG $\mathcal{G}$ if, for any variable $X \in \mathbf{V}$,

$$X \perp\!\!\!\perp Nd(\{X\}) \setminus Pa(\{X\}) \mid Pa(\{X\}) \setminus \{X\} \text{ in } P(\mathbf{v}).$$

Example 1. Consider Fig. 1b. $\{C, D, H\}$ is a c-component, and $\mathcal{C}(H)_{\mathcal{G}^2} = \{C, D, H\}$. Since $\{B, C, D\}$ d-separates H from $\{A, E, F\}$ in $\mathcal{G}^2$, $\mathcal{G}^2$ implies the CI: $H \perp\!\!\!\perp \{A, E, F\} \mid \{B, C, D\}$.

3 The C-component Local Markov Property

In this section, we motivate and introduce the c-component local Markov property for causal DAGs with unobserved confounders. In Sec. 3.1, we demonstrate the limitations of the traditional local Markov property (LMP) when applied to non-Markovian DAGs. In Sec. 3.2, to solve this problem, we present the c-component local Markov property (C-LMP) for semi-Markovian DAGs and establish its equivalence with GMP. In Sec. 3.3, we provide a useful property of C-LMP that makes its CIs amenable to listing.

3.1 A Naive Approach to Testing Non-Markovian Graphs

First, we show the limitations of the well-known LMP (Def. 3) in testing non-Markovian DAGs. For each variable X in a given graph, LMP states that X is independent of its non-descendants conditioning on its parents. Intuitively, the parents of X form a minimal set separating X from its non-descendants.

Example 2. Consider Fig. 1a. The DAG $\mathcal{G}^1$ contains only directed edges; assuming all variables are observed, $\mathcal{G}^1$ is Markovian. LMP invokes 11 CIs for $\mathcal{G}^1$: $A \perp\!\!\!\perp \{U_1, U_2, U_3\}$, $B \perp\!\!\!\perp \{U_1, U_2, U_3\} \mid \{A\}$, $C \perp\!\!\!\perp \{A, E, U_1, U_2\} \mid \{B, U_3\}$, $D \perp\!\!\!\perp \{A, B, E, F, U_1, U_3\} \mid \{C, U_2\}$, $E \perp\!\!\!\perp \{A, C, D, F, H, U_1, U_2, U_3\} \mid \{B\}$, $F \perp\!\!\!\perp \{A, B, E, D, H, U_1, U_2, U_3\} \mid \{C\}$, $H \perp\!\!\!\perp \{A, B, C, E, F, U_2\} \mid \{D, U_1, U_3\}$, $U_1 \perp\!\!\!\perp \{A, B, C, D, E, F, U_3\} \mid \{U_2\}$, $U_2 \perp\!\!\!\perp \{A, B, C, E, F, U_3\}$, $U_3 \perp\!\!\!\perp \{A, B, E, U_1, U_2\}$. All 11 CIs are testable using samples from the distribution $P(a, b, c, d, e, f, h, u_1, u_2, u_3)$. □

LMP fails trivially for semi-Markovian DAGs since, for example, a variable may be connected to a non-descendant by a bidirected edge. One could think to instead apply LMP to the 'unprojected' non-Markovian DAG underlying the given semi-Markovian DAG. The non-Markovian DAG would contain no bidireced edges since the unobserved parents are made explicit. However, LMP does not extend to non-Markovian DAGs either, as we show in the following example.

Example 3. Continuing Ex. 2. Assume we are given the non-Markovian DAG $\mathcal{G}^1$ shown in Fig. 1a. If U_1, U_2 and U_3 are unobserved, only samples from $P(\mathbf{v}) = \int_{u_1, u_2, u_3} P(a, b, c, d, e, f, h, u_1, u_2, u_3) \; du_1 du_2 du_3$ are available, where $\mathbf{V} = \{A, B, C, D, E, F, H\}$ denotes the observed variables. All 11 CIs invoked by LMP for $\mathcal{G}^1$, listed in Ex. 2, require samples from $P(a, b, c, d, e, f, h, u_1, u_2, u_3)$. Hence, none of these CIs can be tested using $P(\mathbf{v})$.

One approach to try salvaging these 11 CIs is to consider only those CIs in which $\{U_1, U_2, U_3\}$ appear before the conditioning bar. In such CIs, $\{U_1, U_2, U_3\}$ can be removed using the decomposition axiom. However, only two of the 11 CIs can be modified in this way, i.e.,

$$E \perp\!\!\!\perp \{A, C, D, F, H\} \mid \{B\}, \tag{1}$$

$$F \perp\!\!\!\perp \{A, B, E, D, H\} \mid \{C\}. \tag{2}$$

These two CIs do not suffice to derive the GMP for $\mathcal{G}^1$. To witness, consider a graph $\mathcal{G}'$ over the same variables as $\mathcal{G}^1$ but with only one edge $H \rightarrow A$. Say we have an observational distribution $P(\mathbf{v})$ faithfully induced by $\mathcal{G}'$. Then, the CIs in

[1]Note that this property is referred to as the *directed local Markov property* in (Lauritzen et al. 1990).

Eqs. (1,2) both hold in $P(\mathbf{v})$. However, $\mathcal{G}^1$ implies that

$$H \perp\!\!\!\perp \{A, E, F\} \mid \{B, C, D\} \tag{3}$$

which does not hold in $P(\mathbf{v})$ since $\mathcal{G}'$ contains an edge $H \to A$. Only testing the two CIs in Eqs. (1,2) would lead to the false conclusion that $P(\mathbf{v})$ is consistent with $\mathcal{G}^1$. As a result, it is insufficient to use only those CIs which invoke $\{U_1, U_2, U_3\}$ outside the conditioning set. □

As in the example, to test a non-Markovian DAG, one can not simply 'filter out' CIs that require conditioning on unobserved variables. This is because such CIs can entail testable CIs over the observed variables. The remaining option is to derive all these entailed CIs using the semi-graphoid axioms, and test those which invoke only observed variables. This is the GMP (Def. 2) of the non-Markovian DAG, which can invoke $\Theta(4^n)$ CIs for a DAG with n observed variables (Prop. C.3.1). This approach fails to exploit any locality in the graph, and requires a prohibitive number of CI tests, many of which are redundant. This suggests the need for alternative compatibility properties for semi-Markovian (equivalently, non-Markovian) DAGs. We next introduce our contribution, the c-component local Markov property.

3.2 C-LMP: A Local Markov Property for Semi-Markovian DAGs

In a semi-Markovian graph, the observed parents of a variable do not suffice to separate it from its non-descendants. Therefore, a surrogate of the parents is needed to restore locality. The construct of a c-component (Def. 1) was introduced for this purpose (Bareinboim et al. 2022), which we explain via an example.

Example 4. Continuing Ex. 2, assume $\{U_1, U_2, U_3\}$ are unobserved in $\mathcal{G}^1$ (Fig. 1a). The second graph $\mathcal{G}^2$ (Fig. 1b) is the semi-Markovian projection of $\mathcal{G}_1$. Note that the conditional independence $H \perp\!\!\!\perp \{A, B, C, E, F\} \mid \{D, U_1, U_3\}$ cannot be tested from the data since $\{U_1, U_3\}$ are not observed. This means that a different conditioning set is needed to make H independent of its observed non-descendants.

One might condition on the observed descendants of $\{U_1, U_3\}$ that are closest to U_1, U_3, i.e., $\{C, D\}$. These variables are not separable from H without conditioning on U_1, U_2 or U_3, which is not an option. $\{C, D\}$ have bidirected edges to H in $\mathcal{G}^2$, the semi-Markovian projection of $\mathcal{G}^1$. $\{C, D\}$ are now active on any paths on which $\{C, D\}$ they are colliders: for instance, on the paths $E \leftarrow B \to C \leftarrow U_3 \to H$ and $A \to B \to C \leftarrow U_3 \to H$. To block some of these paths, we also condition on the (remaining) observed parents of $\{C, D\}$, i.e., $\{B\}$. Firstly, conditioning on $\{C, D\}$ already makes B and its ancestors active on any paths where they are colliders; secondly, B is connected to H when conditioning on $\{C, D\}$. Therefore, conditioning on $\{B\}$ does not introduce any new active paths to X. Conditioning on $\{B\}$ additionally blocks paths to H containing B on which B is not a collider. Therefore, we have the conditioning set $Pa(\mathbf{C}) \setminus \{H\} = \{B, C, D\}$. The CI over observables $H \perp\!\!\!\perp \{A, E, F\} \mid \{B, C, D\}$ is thus derived. □

Ex. 4 is relatively simple since the c-component of H is used to generate the given CI. However, the c-components of a variable do not always give rise to CIs.

Example 5. Consider, as an example DAG, a bidirected path of the form $V_1 \leftrightarrow V_2 \cdots \leftrightarrow V_n$ on variables $\mathbf{V}$. For each V_i, the c-component including V_i is the entire graph. Therefore, conditioning on the c-component results in the 'vacuous' CI: $V_i \perp\!\!\!\perp \emptyset \mid \mathbf{V} \setminus V_i$. Clearly, from this set of vacuous CIs, we cannot derive non-vacuous CIs encoded the graph, such as those of the form $V_i \perp\!\!\!\perp \{V_j\}, \forall i, j$ s.t. $|i - j| > 1$ (e.g., $V_1 \perp\!\!\!\perp \{V_3\}$). □

A useful insight due to (Richardson 2003) is that subsets of a variable's c-component can give rise to distinct 'surrogates' for its parents and hence distinct CIs. This is because conditioning on a certain variable in a c-component closes some paths while opening others. We generalize c-components to *ancestral c-components* to define these 'surrogates.'[2]

Definition 4. (Ancestral C-component (AC)) Given a causal graph $\mathcal{G}$ and a consistent ordering $\mathbf{V}^{\prec}$, let X be a variable in $\mathbf{V}^{\prec}$. A set of variables $\mathbf{C}$ is said to be an ancestral c-component relative to X if there exists an ancestral set $\mathbf{S} \subseteq \mathbf{V}^{\leq X}$ containing X such that $\mathcal{C}(X)_{\mathcal{G}_\mathbf{S}} = \mathbf{C}$. The collection of all such $\mathbf{C}$ is denoted:

$$\mathcal{AC}_X = \{\mathbf{C} \mid \mathbf{C} \text{ is an ancestral c-component relative to } X\}.$$

Unlike c-components, there may be many ancestral c-components with respect to a given variable.

Example 6. Consider the graph $\mathcal{G}$ in Fig. 2 with ordering $A \prec B \prec \cdots \prec X \prec J \prec K$. For the variable X, $\{X\}$ is an AC relative to X induced by the ancestral set $\mathbf{S} = \{X\}$; $\{B, X\}$ is an AC relative to X induced by the ancestral set $\mathbf{S} = \{B, C, D, E, X\}$. $\{X, A, D, E\}$ is not an AC relative to X since the exclusion of B and/or H disconnects the variables in question. For the variable J, $\{J\}$ is not an AC relative to J since it excludes the ancestor X to which J is connected by a bidirected edge; $\{X, J\}$ is an AC induced by the ancestral set $\{X, J\}$. □

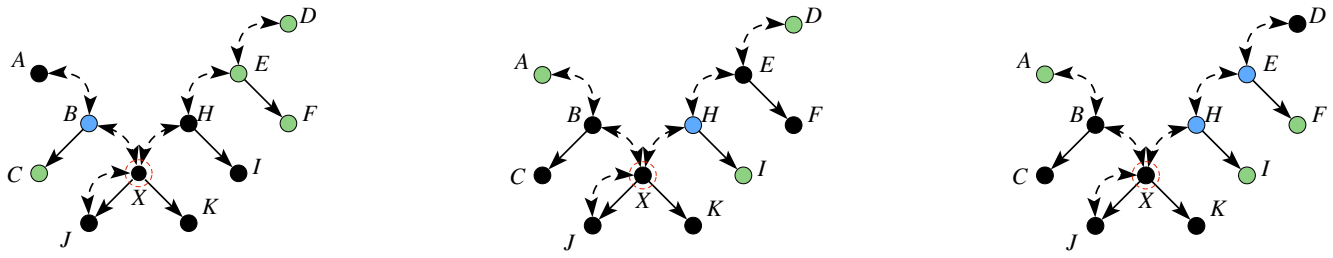
We use ACs to define the c-component local Markov property, which generalizes LMP to semi-Markovian DAGs using this new notion of local independence.

Definition 5. (The C-component Local Markov Property (C-LMP)) A probability distribution $P(\mathbf{v})$ over a set of variables $\mathbf{V}$ is said to satisfy the c-component local Markov property for a causal graph $\mathcal{G}$ with respect to the consistent ordering $\mathbf{V}^{\prec}$, if, for any variable $X \in \mathbf{V}^{\prec}$ and ancestral c-component $\mathbf{C} \in \mathcal{AC}_X$ relative to X,

$$X \perp\!\!\!\perp \mathbf{S}^+ \setminus Pa(\mathbf{C}) \mid (Pa(\mathbf{C}) \setminus \{X\}) \text{ in } P(\mathbf{v}), \text{ where}$$
$$\mathbf{S}^+ = \mathbf{V}^{\leq X} \setminus De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})).$$

Example 7. Continuing Ex. 6. We give a few examples of CIs invoked by C-LMP for the variable X.

[2]Ancestral c-components can be shown to be equivalent to *ancestrally closed districts* as defined in (Richardson 2009). We thank Robin Evans for bringing this to our attention.



(a) X is separated from C but not A when conditioning on B.

(b) X is separated from A but not C when not conditioning on B.

(c) X is separated from F, I but not D when conditioning on $\{H, E\}$.

Figure 2: Three ACs relative to the variable X in the (same) causal DAG $\mathcal{G}$. Assume an ordering $A \prec B \prec \cdots \prec X \prec J \prec K$. The ACs relative to X (excluding $\{X\}$ itself), shown in blue, separate it from the variables shown in green.

1. The AC $\mathbf{C} = \{X, B\}$ gives the CI $X \perp\!\!\!\perp \{C, D, E, F\} \mid \{B\}$ (Fig. 2a), since
$$Pa(\mathbf{C}) = Pa(\{X, B\}) = \{X, B\}$$
$$\begin{aligned}\mathbf{S}^+ &= \mathbf{V}^{\leq X} \setminus De(Sp(\{X, B\} \setminus Pa(\{X, B\}))) \\ &= \{A, B, C, D, E, F, H, I, X\} \setminus De(\{A, H\}) \\ &= \{A, B, C, D, E, F, H, I, X\} \setminus \{A, H, I\} \\ &= \{B, C, D, E, F, X\}\end{aligned}$$
2. The AC $\mathbf{C} = \{X, H\}$ gives the CI $X \perp\!\!\!\perp \{A, D, I\} \mid \{H\}$ (Fig. 2b), since
$$Pa(\mathbf{C}) = Pa(\{X, H\}) = \{X, H\}$$
$$\begin{aligned}\mathbf{S}^+ &= \mathbf{V}^{\leq X} \setminus De(Sp(\{X, H\} \setminus Pa(\{X, H\}))) \\ &= \{A, B, C, D, E, F, H, I, X\} \setminus De(\{B, E\}) \\ &= \{A, B, C, D, E, F, H, I, X\} \setminus \{B, C, E, F\} \\ &= \{A, D, H, I, X\}.\end{aligned}$$
3. The AC $\mathbf{C} = \{X, H, E\}$ gives the CI $X \perp\!\!\!\perp \{A, F, I\} \mid \{H, E\}$ (Fig. 2c), since
$$Pa(\mathbf{C}) = Pa(\{X, H, E\}) = \{X, H, E\}$$
$$\begin{aligned}\mathbf{S}^+ &= \mathbf{V}^{\leq X} \setminus De(Sp(\{X, H, E\} \setminus Pa(\{X, H, E\}))) \\ &= \{A, B, C, D, E, F, H, I, X\} \setminus De(\{B, D\}) \\ &= \{A, B, C, D, E, F, H, I, X\} \setminus \{B, C, D, E\} \\ &= \{A, E, F, H, I, X\}\end{aligned}$$

□

As a sanity check, let us examine the CIs C-LMP implies for a Markovian DAG $\mathcal{G}$, where all c-components are singletons. There is exactly one AC $\mathbf{C} = \{X\}$ relative to a given variable X. Moreover, $Pa(\mathbf{C}) = Pa(\{X\})$, $Sp(\{X\}) = \emptyset$ and $\mathbf{S}^+ = \mathbf{V}^{\leq X} \setminus De(\emptyset) = \mathbf{V}^{\leq X}$. Therefore, the CI invoked by C-LMP for X is

$$X \perp\!\!\!\perp \mathbf{V}^{\leq X} \setminus Pa(\{X\}) \mid Pa(\{X\}) \setminus \{X\} \tag{4}$$

Thus, C-LMP reduces to the local well-numbering Markov property (Lauritzen et al. 1990) for a Markovian DAG $\mathcal{G}$.[3]

[3]The local well-numbering Markov property was shown to be equivalent to LMP in (Lauritzen et al. 1990). A subtle difference is that LMP tests the independence of X from its all non-descendants, not just $\mathbf{V}^{\leq X}$ for a given ordering $\mathbf{V}^{\prec}$.

In semi-Markovian DAGs, c-components are not necessarily singletons. Comparing the CIs invoked by LMP and C-LMP for a given variable X, we see that C-LMP generalizes two concepts:

1. The conditioning set $Pa(\{X\}) \setminus \{X\}$ stated in LMP is replaced with $Pa(\mathbf{C}) \setminus \{X\}$ in C-LMP, using an AC $\mathbf{C}$ relative to X.
2. The conditioning set $Pa(\mathbf{C}) \setminus \{X\}$ renders X independent of $\mathbf{S}^+ \setminus Pa(\mathbf{C})$ where $\mathbf{S}^+ = \mathbf{V}^{\leq X} \setminus De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C}))$, as stated by C-LMP. The set $\mathbf{S}^+ \setminus Pa(\mathbf{C})$ replaces the set $Nd(\{X\}) \setminus Pa(\{X\})$ in LMP.

In Ex. 4, we gave intuition for the generalised conditioning set $Pa(\mathbf{C}) \setminus \{X\}$ (Case 1). Next, we explain the construction of $\mathbf{S}^+$ (Case 2) used to compute the maximal set of variables in $\mathbf{V}^{\leq X}$ that are independent of X given $Pa(\mathbf{C}) \setminus \{X\}$. Consider what happens to a variable $Y \in \mathbf{V}^{\leq X} \setminus Pa(\mathbf{C})$ when conditioning on $Pa(\mathbf{C}) \setminus \{X\}$.

- If Y is a descendant (or an ancestor) of some node $W \in Pa(\mathbf{C})$, we have a directed path π from W to Y (or vice-versa). Conditioning on $Pa(\mathbf{C}) \setminus \{X\}$ blocks π (since $Y \notin Pa(\mathbf{C})$), and hence any path from X to Y which contains π as a sub-path. For example, in Fig. 2a, taking $\mathbf{C} = \{X, B\}$, $W = B$ and $Y = C$, conditioning on $\{B\}$ blocks the path $X \leftrightarrow B \rightarrow C$.
- If Y is connected by a bidirected path to some node in $\mathbf{C}$, but Y is not in $Sp(\mathbf{C})$, then some node $V \in Sp(\mathbf{C}) \setminus Pa(\mathbf{C})$ 'intercepts' this path, i.e., V is a closed collider and thus blocks the path from X to Y. For example, in Fig. 2a, taking $\mathbf{C} = \{X, B\}$, $V = H$ and $Y = E$, H blocks the path $X \leftrightarrow H \leftrightarrow E$.
- If Y is in $Sp(\mathbf{C}) \setminus Pa(\mathbf{C})$, is an active bidirected path from X to Y when conditioning on $\mathbf{C} \setminus \{X\}$. For example, in Fig. 2a, taking $\mathbf{C} = \{X, B\}$ and $Y = A$, conditioning on $\{B\}$ opens the path $X \leftrightarrow B \leftrightarrow A$.

Analogous to how, for a given variable X, different conditioning sets give rise to different CIs from X, different ACs also give rise to different CIs from X. The upshot of defining ACs is that they carve out a relatively small set of CIs (commonly known as a 'basis' (Bareinboim et al. 2022)) from which all CIs encoded in the given graph can be derived.

The main result of this section, given below, establishes that GMP and C-LMP are equivalent.

Theorem 1 (Equivalence of C-LMP and GMP)**.** *Let $\mathcal{G}$ be a causal graph and $\mathbf{V}^{\prec}$ a consistent ordering. A probability distribution over $\mathbf{V}$ satisfies the global Markov property for $\mathcal{G}$ if and only if it satisfies the c-component local Markov property for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$.*

As a corollary of Thm. 1, we can conclude that C-LMP is equivalent to Richardson's ordered local Markov property (Richardson 2003), since the latter is equivalent to GMP (Richardson 2003, Thm. 2, Section 3.1).

Corollary 1 (Equivalence of C-LMP and the Ordered Local Markov Property)**.** *Let $\mathcal{G}$ be a causal graph and $\mathbf{V}^{\prec}$ a consistent ordering. A probability distribution over $\mathbf{V}$ satisfies the ordered local Markov property (Richardson 2003) for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$ if and only if it satisfies the c-component local Markov property for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$.*

In Appendix B, we further develop the connection between C-LMP and the ordered local Markov property. In fact, in Thm. B.2.1, we show that these two properties induce the exact same set of CIs for a given DAG and a consistent ordering. Thm. B.2.1 thus provides another way to obtain Thm. 1 as a corollary.

The equivalence of C-LMP and GMP means that the CIs invoked by C-LMP for a given causal DAG can be used to test the DAG against observational data.

3.3 Uniqueness Property of C-LMP

By definition, each CI invoked by C-LMP is generated from an AC. We further show that each CI can be generated from exactly one AC.

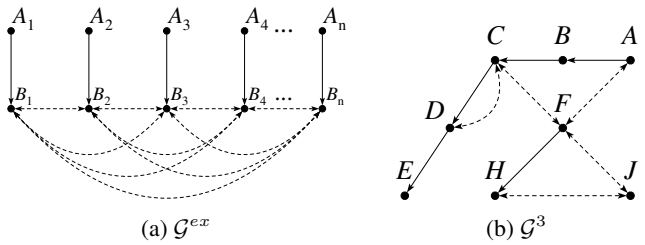
Theorem 2 (Unique AC for each CI Invoked by C-LMP)**.** *Let $\mathcal{G}$ be a causal graph, $\mathbf{V}^{\prec}$ a consistent ordering, and X a variable in $\mathbf{V}^{\prec}$. For every conditional independence relation invoked by the c-component local Markov property of the form $X \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z}$, there is exactly one ancestral c-component $\mathbf{C} \in \mathcal{AC}_X$ such that $\mathbf{W} = \mathbf{V}^{\leq X} \setminus ((De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C}))) \cup Pa(\mathbf{C}))$ and $\mathbf{Z} = Pa(\mathbf{C}) \setminus \{X\}$.*

The one-to-one correspondence between ACs and CIs invoked by C-LMP allows us to give bounds on the latter number that are tight in the exponent.

Proposition 1 (Number of CIs Invoked by C-LMP)**.** Given a causal graph $\mathcal{G}$ and a consistent ordering $\mathbf{V}^{\prec}$, let n and $s \leq n$ denote the number of variables and the size of the largest c-component in $\mathcal{G}$ respectively. Then, the c-component local Markov property for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$ invokes $O(n2^s)$ conditional independencies implied by $\mathcal{G}$ over $\mathbf{V}$. Moreover, there exists a graph $\mathcal{G}$ and a consistent ordering $\mathbf{V}^{\prec}$ for which the property induces $\Omega(2^n)$ conditional independencies.

This result shows that C-LMP offers an exponential improvement on the $\Theta(4^n)$ CIs invoked by GMP. However, C-LMP can still invoke an exponential number of CIs. For example, in $\mathcal{G}^{ex}$ (Fig. 3a) with $2n$ nodes, there are $2^n + (n-3)$ CIs invoked by C-LMP.

The main upshot of the one-to-one correspondence between ACs and CIs invoked by C-LMP is that to list such CIs, it suffices to enumerate ACs. We study the problem of listing CIs in the next section.



(a) $\mathcal{G}^{ex}$ (b) $\mathcal{G}^3$

Figure 3: (a) An example showing that C-LMP may invoke an exponential number of CIs. (b) A causal graph used to show the execution of LISTCI in Ex. 9.

4 Listing CIs

Our goal in this section is to develop an algorithm that lists CIs invoked by C-LMP. In the worst case, there may exist exponentially many such CIs, requiring exponential time to list them all. In such cases, we look for algorithms that run in polynomial delay (Johnson, Yannakakis, and Papadimitriou 1988). Poly-delay algorithms output the first solution (or indicate none is available) in poly-time, and take poly-time to output each consecutive solution.

However, not all CIs invoked by C-LMP are useful for model testing. C-LMP invokes some 'vacuous' CIs of the form $X \perp\!\!\!\perp \emptyset \mid \mathbf{Z}$, which do not need testing. Therefore, we constrain the problem by requiring that we list only *non-vacuous* CIs, as defined below.

Definition 6 (Vacuous CI and Admissible AC (AAC))**.** Given a conditional independence relation invoked by C-LMP of the form $X \perp\!\!\!\perp \mathbf{W} \mid Pa(\mathbf{C}) \setminus \{X\}$, where $\mathbf{W} = \mathbf{S}^{+} \setminus Pa(\mathbf{C})$ (by Def. 5), if $\mathbf{W} \neq \emptyset$, the conditional independence relation is said to be non-vacuous and $\mathbf{C}$ is said to be an admissible ancestral c-component relative to X.

Example 8. Consider the causal graph $\mathcal{G}^3$ (Fig. 3b). The AC $\{J\}$ relative to J is admissible. Given $\mathbf{S}^{+} = \mathbf{V} \setminus \{F, H\}$, we have $\mathbf{W} = \mathbf{S}^{+} \setminus \{J\} = \{A, B, C, D, E\}$. However, the AC $\{F, J\}$ relative to J is not admissible. Since $\mathbf{S}^{+} = \{F, J\}$, $\mathbf{W} = \mathbf{S}^{+} \setminus \{F, J\} = \emptyset$. □

Listing only non-vacuous CIs is important since C-LMP may invoke exponentially many vacuous CIs. To witness, consider a bidirected clique on n nodes such that no two variables are independent of each other given any conditioning set. Every set $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X\}$ forms an AC, resulting in $\Omega(2^n)$ vacuous CIs (see Ex. D.3.1 in Appendix D.3 for details).

Our bounds on the number of CIs invoked by C-LMP are also tight for the number of non-vacuous CIs (Prop. 1). We develop the algorithm LISTCI (Alg. 1) to list all non-vacuous CIs invoked by C-LMP in poly-delay.

Example 9. Consider the causal graph $\mathcal{G}^3$ (Fig. 3b) with $\mathbf{V}^{\prec} = \{A, B, C, D, E, F, H, J\}$. LISTCI$(\mathcal{G}^3, \mathbf{V}^{\prec})$ lists 11 non-vacuous CIs invoked by C-LMP: $C \perp\!\!\!\perp \{A\} \mid \{B\}$, $D \perp\!\!\!\perp \{A\} \mid \{B, C\}$, $E \perp\!\!\!\perp \{A, B, C\} \mid \{D\}$, $F \perp\!\!\!\perp \{B\} \mid \{A\}$, $F \perp\!\!\!\perp \{E\} \mid \{A, B, C, D\}$, $H \perp\!\!\!\perp \{A, B, C, D, E\} \mid \{F\}$, $J \perp\!\!\!\perp$

Algorithm 1: LISTCI ($\mathcal{G}, \mathbf{V}^{\prec}$)

1: **Input:** $\mathcal{G}$ a causal diagram; $\mathbf{V}^{\prec}$ an ordering consistent with $\mathcal{G}$.
2: **Output:** Listing non-vacuous CIs invoked by C-LMP for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$.
3: **for** each $X \in \mathbf{V}^{\prec}$ **do**
4: $\quad \mathbf{I} \leftarrow \mathcal{C}(X)_{\mathcal{G}_{An(\{X\})}}, \mathbf{R} \leftarrow \mathcal{C}(X)_{\mathcal{G}_{\mathbf{V}\leq X}}$
5: $\quad$ LISTCIX($\mathcal{G}_{\mathbf{V}\leq X}, X, \mathbf{V}^{\leq X}, \mathbf{I}, \mathbf{R}$)

$\{A, B, C, D, E\}$, $J \perp\!\!\!\perp \{B\} \mid \{A, F\}$, $J \perp\!\!\!\perp \{B\} \mid \{A, F, H\}$, $J \perp\!\!\!\perp \{E\} \mid \{A, B, C, D, F\}$, $J \perp\!\!\!\perp \{E\} \mid \{A, B, C, D, F, H\}$. After, LISTCI terminates as there are no more non-vacuous CIs. □

4.1 Listing CIs for a Given Variable

The algorithm LISTCI iterates over each variable $X \in \mathbf{V}^{\prec}$ and lists all non-vacuous CIs invoked by C-LMP for X. By Defs. 5, 6 and Thm. 2, listing non-vacuous CIs reduces to enumerating AACs. In this section, we show how to enumerate AACs relative to a given variable $X \in \mathbf{V}^{\prec}$ using the procedure LISTCIX (Alg. 2).

LISTCIX adopts a divide-and-conquer strategy similar to the algorithm of (Takata 2010). LISTCIX implicitly constructs a binary search tree for X using a depth-first approach. Tree nodes of the form $\mathcal{N}(\mathbf{I}', \mathbf{R}')$ represents the collection of all AACs $\mathbf{C}$ with $\mathbf{I}' \subseteq \mathbf{C} \subseteq \mathbf{R}'$. The top-level call of LISTCIX, at the root node $\mathcal{N}(\mathbf{I}, \mathbf{R})$, represents all AACs $\mathbf{C}$ relative to X. This is due to the construction on line 4 of LISTCI so that $\mathbf{I}$ is contained in and $\mathbf{R}$ contains all possible AACs relative to X. Thus, the top-level call can generate all CIs for X.

Subsequent recursive calls expand this tree by shrinking the range $\mathbf{I}' \subseteq \mathbf{R}'$ one variable at a time. One requirement of the poly-delay property is that each AAC should appear exactly once in the enumeration. To expand the tree from $\mathcal{N}(\mathbf{I}', \mathbf{R}')$, LISTCIX constructs two 'disjoint' children (lines 10-11: a chosen variable $S \in \mathbf{V}^{\prec}$ cannot be in any AAC from the left child but must be in every AAC from the right child.

As another requirement of the poly-delay property, we expand the tree from a node $\mathcal{N}(\mathbf{I}', \mathbf{R}')$ if and only if the expansion is guaranteed to produce a non-vacuous CI. Equivalently, there must exist at least one AAC $\mathbf{C}$ such that $\mathbf{I}' \subseteq \mathbf{C} \subseteq \mathbf{R}'$. If there is no such $\mathbf{C}$, we prune the tree and back-track to the previous tree node. To perform this check for the existence of an AAC in poly-time, LISTCIX calls the function FINDAAC (Alg. 3). We explain FINDAAC in the next subsection.

Finally, a leaf node is reached when $\mathbf{I} = \mathbf{R}$. LISTCIX outputs a non-vacuous CI generated from the AAC $\mathbf{C} = \mathbf{I}$ using Def. 5.

Example 10. Expanding Ex. 9 to demonstrate the construction of the search tree $\mathcal{T}^3$ (Fig. 4) generated by running LISTCI($\mathcal{G}^3, \mathbf{V}^{\prec}$) for $X = J$. With $\mathbf{I} = \{J\}$ and $\mathbf{R} = \{A, C, D, F, H, J\}$ constructed on line 4, the initial search starts from the root node $\mathcal{N}(\mathbf{I}, \mathbf{R})$ on line 5. On line 3 of LISTCIX, FINDAAC returns $\{J\}$. With $S = F$ and

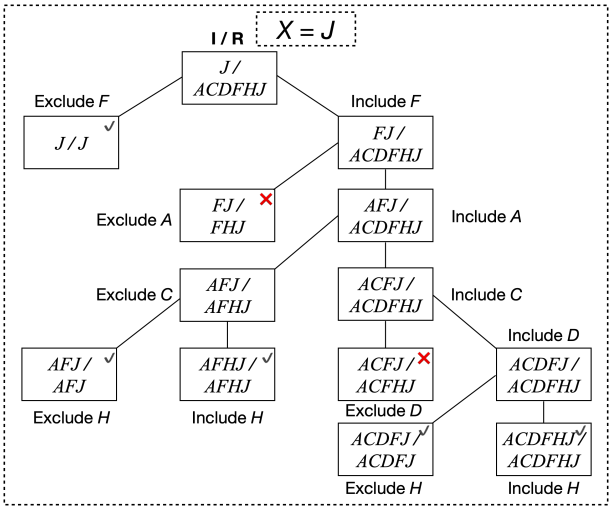


Figure 4: $\mathcal{T}^3$ a search tree illustrating the running of LISTCI in Ex. 9 for $X = J$.

$\mathbf{R}' = \{J\}$, the recursive call LISTCIX($\mathcal{G}^3, J, \mathbf{V}^{\prec}, \mathbf{I}, \mathbf{R}'$) is made at line 10, spawning a child $\mathcal{N}_1(\{J\}, \{J\})$. The search continues from $\mathcal{N}_1$. FINDAAC returns $\{J\}$. $\mathcal{N}_1$ is a leaf node, and LISTCIX outputs a CI: $J \perp\!\!\!\perp \{A, B, C, D, E\}$ on line 6. The rest of the search tree for J is shown in $\mathcal{T}^3$. The full set of search trees is shown in Fig. D.3.1 in Appendix D.3. □

4.2 Finding an AAC

In this section, we address the following subproblem, needed for LISTCIX to run in poly-delay: given a variable $X \in \mathbf{V}^{\prec}$, and two ACs $\mathbf{I}, \mathbf{R}$ relative to X, how do we find an AAC $\mathbf{C}$ such that $\mathbf{I} \subseteq \mathbf{C} \subseteq \mathbf{R}$ (or indicate that there is none) in poly-time?

The poly-time constraint on solving this subproblem rules out the brute-force approach: namely, iterating over all subsets $\mathbf{C}$ such that $\mathbf{I} \subseteq \mathbf{C} \subseteq \mathbf{R}$ until we find some $\mathbf{C}$ that is an AAC (or conclude that there is none). We find that this exponential search is not necessary. The key idea behind our solution is to reduce this search to a verification of whether a particular AAC is admissible.

To elaborate, assume there exists an AAC $\mathbf{C}$ such that $\mathbf{I} \subseteq \mathbf{C} \subseteq \mathbf{R}$. It is possible that $\mathbf{C} = \mathbf{I}$, in which case we are done. Otherwise, consider what it means for $\mathbf{I}$ not to be admissible. When conditioning on $Pa(\mathbf{I}) \setminus \{X\}$, every variable in $\mathbf{V}^{\leq X}$ has an active path to X. This means that every such variable (outside the conditioning set) is a descendant of $Sp(\mathbf{I}) \setminus Pa(\mathbf{I})$ (Def. 5). Since $\mathbf{C}$ is admissible, then there must be some descendant of $Sp(\mathbf{I}) \setminus Pa(\mathbf{I})$, say D, which has an active path to X when conditioning on $Pa(\mathbf{I}) \setminus \{X\}$ but not when conditioning on $Pa(\mathbf{C}) \setminus \{X\}$. We show that the AAC $\mathbf{C}$ can exist if and only if there exists *any* set $\mathbf{Z}$ separating X from some such D, with the restriction that $Pa(\mathbf{I}) \setminus \{X, D\} \subseteq \mathbf{Z} \subseteq Pa(\mathbf{R}) \setminus \{X, D\}$. $\mathbf{Z}$ need not be an AAC. We can check if such $\mathbf{Z}$ exists (line 6) in poly-time using the function FINDSEPARATOR (Fig. C.2.2 in Appendix C.2).

Algorithm 2: LISTCIX ($\mathcal{G}_{\mathbf{V}\leq x}, X, \mathbf{V}^{\leq X}, \mathbf{I}, \mathbf{R}$)

Input: $\mathcal{G}_{\mathbf{V}\leq x}$ a causal diagram; X a variable; $\mathbf{V}^{\leq X}$ an ordering consistent with $\mathcal{G}$; $\mathbf{I}$ and $\mathbf{R}$ ACs relative to X.
Output: Listing non-vacuous CIs invoked by C-LMP associated with X and AACs $\mathbf{C}$ under the constraint $\mathbf{I} \subseteq \mathbf{C} \subseteq \mathbf{R}$.
if FINDAAC($\mathcal{G}_{\mathbf{V}\leq x}, X, \mathbf{V}^{\leq X}, \mathbf{I}, \mathbf{R}$) $\neq \perp$ **then**
    if $\mathbf{I} = \mathbf{R}$ **then**
        $\mathbf{S}^+ \leftarrow \mathbf{V}^{\leq X} \setminus De(Sp(\mathbf{I}) \setminus Pa(\mathbf{I}))$
        Output $X \perp\!\!\!\perp \mathbf{S}^+ \setminus Pa(\mathbf{I}) \mid Pa(\mathbf{I}) \setminus \{X\}$
        return
    $\mathbf{T} \leftarrow \mathbf{R} \cap (Sp(\mathbf{I}) \setminus \mathbf{I}), S \leftarrow$ Any node in $\mathbf{T}$
    $\mathbf{I}' \leftarrow \mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{I}\cup\{S\})}}, \mathbf{R}' \leftarrow \mathcal{C}(X)_{\mathcal{G}_{\mathbf{R}\setminus De(\{S\})}}$
    LISTCIX($\mathcal{G}_{\mathbf{V}\leq x}, X, \mathbf{V}^{\leq X}, \mathbf{I}, \mathbf{R}'$)
    LISTCIX($\mathcal{G}_{\mathbf{V}\leq x}, X, \mathbf{V}^{\leq X}, \mathbf{I}', \mathbf{R}$)

Example 11. Expanding Ex. 10 to illustrate the usage of FINDAAC. Let $X = J$, $\mathbf{V}^{\prec} = \mathbf{V}^{\leq J}$, $\mathbf{I} = \{J\}$, and $\mathbf{R} = \{A, C, D, F, H, J\}$. FINDAAC($\mathcal{G}^3, J, \mathbf{V}^{\leq J}, \mathbf{I}, \mathbf{R}$) returns $\mathbf{C} = \{J\}$ since there exists an AAC $\mathbf{C}$ relative to J with $\mathbf{I} \subseteq \mathbf{C} \subseteq \mathbf{R}$. With $\mathbf{I} = \{F, J\}$ and $\mathbf{R} = \{F, H, J\}$, FINDAAC($\mathcal{G}^3, J, \mathbf{V}^{\leq J}, \mathbf{I}, \mathbf{R}$) returns $\perp$ since none of the ACs $\mathbf{C}$ relative to J with $\mathbf{I} \subseteq \mathbf{C} \subseteq \mathbf{R}$ are admissible. □

Lemma 1 (Correctness of FINDAAC)**.** *Given a causal graph $\mathcal{G}$, a consistent ordering $\mathbf{V}^{\prec}$, and a variable $X \in \mathbf{V}^{\prec}$, let $\mathbf{I}, \mathbf{R}$ be ancestral c-components relative to X such that $\mathbf{I} \subseteq \mathbf{R}$.* FINDAAC*($\mathcal{G}_{\mathbf{V}\leq x}, X, \mathbf{V}^{\leq X}, \mathbf{I}, \mathbf{R}$) outputs an admissible ancestral c-component $\mathbf{C}$ relative to X such that $\mathbf{I} \subseteq \mathbf{C} \subseteq \mathbf{R}$ if such a $\mathbf{C}$ exists, and $\perp$ otherwise.*

Lemma 2 (Correctness of LISTCIX)**.** LISTCIX *($\mathcal{G}_{\mathbf{V}\leq x}, X, \mathbf{V}^{\leq X}, \mathbf{I}, \mathbf{R}$) enumerates all and only all non-vacuous conditional independence relations invoked by the c-component local Markov property associated with X and admissible ancestral c-components $\mathbf{C}$ relative to X where $\mathbf{I} \subseteq \mathbf{C} \subseteq \mathbf{R}$. Further,* LISTCIX *runs in $O(n^2(n+m))$ delay where n and m represent the number of nodes and edges in $\mathcal{G}$, respectively.*

Our results are summarized in the following theorem, which provides the soundness, completeness, and poly-delay complexity of the proposed algorithm.

Theorem 3 (Correctness of LISTCI)**.** *Let $\mathcal{G}$ be a causal graph and $\mathbf{V}^{\prec}$ a consistent ordering.* LISTCI*($\mathcal{G}, \mathbf{V}^{\prec}$) enumerates all and only all non-vacuous conditional independence relations invoked by the c-component local Markov property in $O(n^2(n+m))$ delay where n and m represent the number of nodes and edges in $\mathcal{G}$, respectively.*

5 Experiments

In this section, we first demonstrate the runtime of LISTCI on benchmark DAGs of up to 100 nodes from the `bnlearn` repository (Scutari 2010). Next, we apply LISTCI to model testing on a real-world protein signaling dataset with an expert-provided graph (Sachs et al. 2005). Third, we provide analysis of the total number of non-vacuous CIs invoked by C-LMP, using LISTCI for the analysis. The details of the three experiments are shown in Appendix F.

Algorithm 3: FINDAAC ($\mathcal{G}_{\mathbf{V}\leq x}, X, \mathbf{V}^{\leq X}, \mathbf{I}, \mathbf{R}$)

Input: $\mathcal{G}_{\mathbf{V}\leq x}$ a causal diagram; X a variable; $\mathbf{V}^{\leq X}$ an ordering consistent with $\mathcal{G}$; $\mathbf{I}$ and $\mathbf{R}$ ACs relative to X.
Output: An AAC $\mathbf{C}$ relative to X under the constraint $\mathbf{I} \subseteq \mathbf{C} \subseteq \mathbf{R}$, if such $\mathbf{C}$ exists; $\perp$ otherwise.
if ISADMISSIBLE($\mathcal{G}_{\mathbf{V}\leq x}, X, \mathbf{V}^{\leq X}, \mathbf{I}$) **then**
    return $\mathbf{I}$
for each $D \in De(Sp(\mathbf{I}) \setminus Pa(\mathbf{I}))$ **do**
    $\mathbf{Z} \leftarrow$ FINDSEPARATOR($\mathcal{G}_{\mathbf{V}\leq x}, \{X\}, \{D\}, Pa(\mathbf{I}), Pa(\mathbf{R})$)
    if $\mathbf{Z} \neq \perp$ **then**
        return $\mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{I}\cup\mathbf{Z})}}$
return $\perp$

Experiment 1 (Comparison of LISTCI with other algorithms). We compare the runtime of LISTCI with two baselines: LISTGMP (Fig. E.0.1 in Appendix E) and LISTCIBF (Alg. B.1.1 in Appendix B.1)[4]. LISTGMP lists all CIs invoked by GMP (Def. 2); LISTCIBF iterates over ancestral sets to list CIs invoked by the ordered local Markov property (Richardson 2003). The algorithms were run on DAGs that describe real-world scenarios from the `bnlearn` repository. Since the graphs are Markovian, non-Markovian graphs were generated by randomly assigning $U\%$ of nodes to be unobserved for $U \in \{0, 10, 20, \ldots, 90\}$. For each U, we generated 10 random samples. For a given graph, algorithm, and U, if any one sample timed out (> 1 hour), no further samples are tested. Fig. 5 shows the average runtime of the algorithms, with further details in Fig. F.1.1.

The results corroborate our theoretical conclusion that LISTCI outperforms the other algorithms. For LISTGMP, the algorithm did not timeout on graphs with $n < 10$ nodes. For LISTCIBF, we have mixed results. The algorithm did not time out for some graphs with up to $n = 35$ nodes, but there were other graphs with $n = 25$ where the algorithm did time out. For LISTCI, the algorithm did not timeout for many graphs up to $n = 80$, but did time out for some graphs with $n = 70$.

Experiment 2 (Application to model testing). A real-world protein signaling dataset (Sachs et al. 2005) has been used to benchmark causal discovery methods (Cundy, Grover, and Ermon 2021; Zantedeschi et al. 2023). The dataset (853 samples) comes with an expert-provided ground-truth DAG (11 nodes, 16 edges). Using LISTCI, we test to what extent this graph is compatible with the available data. We use a kernel-based CI test from the `causal-learn` package (Zheng et al. 2024) with p-value $p = 0.05$.

[4]Our implementation of LISTCIBF can be improved by generating ancestral sets more efficiently. Regardless, we know LISTCI performs better in theory (Sec. 3), and have strong evidence that it is also better in practice to this more efficient implementation of LISTCIBF.

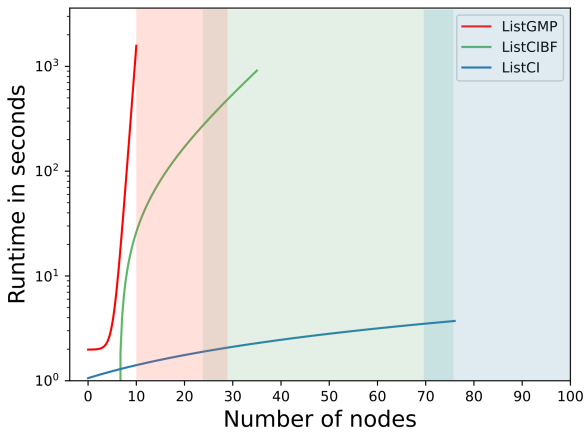


Figure 5: Plot of runtimes of the algorithms LISTGMP, LISTCIBF, and LISTCI on graphs of various sizes. A colored box indicates the interval of n on which the relevant algorithm has timed out on some graphs with n nodes. The y-axis uses a logarithmic scale.

For our chosen topological order, seven out of ten CIs invoked by C-LMP resulted in $p > 0.05$. This suggests the ground-truth DAG may need revision before use as a benchmark for structure learning. The exact local CIs that are violated may guide experts in this revision process.

Experiment 3 (Analysis of C-LMP). We use LISTCI to understand the total number of non-vacuous CIs invoked by C-LMP. Let **CI** denote this number. **CI** is also the number of CIs that need to be tested from a given semi-Markovian causal DAG. Based on experiments with random graphs shown in Appendix F.3, we conclude that the graph topology associated with c-components plays a major role in **CI**. More specifically, two factors related to c-components are of primary interest:

1. $s \leq n$: the size of the largest c-component, and
2. The sparsity of c-components, a proxy for which is the number of bidirected edges.

As we add bidirected edges, while c-components are sparse, **CI** increases exponentially with s, as given by the bound $O(n2^s)$. As c-components become more dense, **CI** decays exponentially with the number of bidirected edges. As an illustrative example, please refer to Fig. F.3.1 and the discussion on Case 1 in Appendix F.3.

6 Conclusions

In this paper, we introduced a new conditional independence property for causal models with unobserved confounders, namely, *the c-component local Markov property* (C-LMP, Def. 5). Given a DAG $\mathcal{G}$, C-LMP identifies a small subset of conditional independence constraints (CIs) that together imply all other CIs encoded in $\mathcal{G}$. We showed that C-LMP is equivalent to the global Markov property (Thm. 1), and that each CI that C-LMP invokes can be generated from a unique ancestral c-component (Thm. 2). Building on this foundation, we developed the first algorithm LISTCI (Alg. 1) capable of listing all CIs invoked by C-LMP in polynomial delay (Thm. 3). Reducing the number of CI tests needed to evaluate a causal model is important as it improves runtime performance and helps mitigate concerns about statistical power and multiple hypothesis testing. We hope our work will help researchers test their causal assumptions using observational data prior to inference.

Acknowledgements

This research is supported in part by the NSF, DARPA, ONR, AFOSR, DoE, Amazon, JP Morgan, and The Alfred P. Sloan Foundation. We thank Robin Evans and the anonymous reviewers for their thoughtful comments.

References

Ankan, A.; and Textor, J. 2022. A Simple Unified Approach to Testing High-Dimensional Conditional Independences for Categorical and Ordinal Data. arXiv:2206.04356.

Bareinboim, E.; Correa, J. D.; Ibeling, D.; and Icard, T. 2022. On Pearl's Hierarchy and the Foundations of Causal Inference. In *Probabilistic and Causal Inference: The Works of Judea Pearl*, 507–556. New York, NY, USA: Association for Computing Machinery, 1st edition.

Bareinboim, E.; and Pearl, J. 2016. Causal inference and the data-fusion problem. *Proceedings of the National Academy of Sciences*, 113(27): 7345–7352.

Cundy, C.; Grover, A.; and Ermon, S. 2021. BCD Nets: Scalable Variational Approaches for Bayesian Causal Discovery. arXiv:2112.02761.

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

Fisher, R. A. 1936. Design of Experiments. *British Medical Journal*, 1(3923): 554.

Geiger, D.; and Meek, C. 1998. Graphical Models and Exponential Families. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*.

Geiger, D.; and Meek, C. 1999. Quantifier Elimination for Statistical Problems. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*.

Geiger, D.; Verma, T. S.; and Pearl, J. 1989. d-Separation: From Theorems to Algorithms. In *Proceedings, 5th Workshop on Uncertainty in AI*, 118–124. ISBN 9780444887382.

Hoover, K. D. 1990. The logic of causal inference: Econometrics and the conditional analysis of causation. *Economics and Philosophy*, 6(2): 207–234.

Hu, Z.; and Evans, R. 2023. Towards standard imsets for maximal ancestral graphs. arXiv:2208.10436.

Jaber, A.; Kocaoglu, M.; Shanmugam, K.; and Bareinboim, E. 2020. Causal Discovery from Soft Interventions with Unknown Targets: Characterization and Learning. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 9551–9561. Vancouver, Canada: Curran Associates, Inc.

Johnson, D. S.; Yannakakis, M.; and Papadimitriou, C. H. 1988. On generating all maximal independent sets. *Information Processing Letters*, 27(3): 119–123.

Kang, C.; and Tian, J. 2009. Markov properties for linear causal models with correlated errors. *The Journal of Machine Learning Research*, 10: 41–70.

King, R. D.; Whelan, K. E.; Jones, F. M.; Reiser, P. G.; Bryant, C. H.; Muggleton, S. H.; Kell, D. B.; and Oliver, S. G. 2004. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971): 247–252.

Kocaoglu, M.; Jaber, A.; Shanmugam, K.; and Bareinboim, E. 2019. Characterization and learning of causal graphs with latent variables from soft interventions. *Advances in Neural Information Processing Systems*, 32.

Kocaoglu, M.; Shanmugam, K.; and Bareinboim, E. 2017. Experimental Design for Learning Causal Graphs with Latent Variables. In *Advances in Neural Information Processing Systems 30*. ISBN 0327-3776, 1850-275X.

Lauritzen, S.; and Sadeghi, K. 2018. Unifying Markov properties for graphical models. *The Annals of Statistics*, 46(5): 2251 – 2278.

Lauritzen, S. L. 1996. *Graphical Models*. Oxford: Clarendon Press.

Lauritzen, S. L.; Dawid, A. P.; Larsen, B. N.; and Leimer, H. G. G. 1990. Independence Properties of Directed Markov Fields. *Networks*, 20(5): 491–505.

Li, A.; Jaber, A.; and Bareinboim, E. 2023. Causal discovery from observational and interventional data across multiple environments. *Advances in Neural Information Processing Systems*, 36: 16942–16956.

Malinsky, D. 2024. A cautious approach to constraint-based causal model selection. arXiv:2404.18232.

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

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufmann.

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

Pearl, J. 1998. Graphs, causality, and structural equation models. *Sociological Methods and Research*, 27(2): 226–284.

Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. New York, NY, USA: Cambridge University Press, 2nd edition.

Pearl, J.; and Meshkat, P. 1999. Testing regression models with fewer regressors. In Heckerman, D.; and Whittaker, J., eds., *Artificial Intelligence and Statistics 99*, 255–259. San Francisco, CA: Morgan Kaufmann.

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

Richardson, T.; and Spirtes, P. 2002. Ancestral graph Markov models. *Ann. Statist.*, 30(4): 962–1030.

Richardson, T. S. 2009. A factorization criterion for acyclic directed mixed graphs. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, 462–470. Arlington, Virginia, USA: AUAI Press. ISBN 9780974903958.

Robins, J. M.; Hernan, M. A.; and Brumback, B. 2000. Marginal structural models and causal inference in epidemiology.

Rotmensch, M.; Halpern, Y.; Tlimat, A.; Horng, S.; and Sontag, D. 2017. Learning a health knowledge graph from electronic medical records. *Scientific reports*, 7(1): 5994.

Sachs, K.; Perez, O.; Pe'er, D.; Lauffenburger, D. A.; and Nolan, G. P. 2005. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721): 523–529.

Scutari, M. 2010. Learning Bayesian Networks with the bnlearn R Package. *Journal of Statistical Software*, 35(3): 1–22.

Shachter, R. D. 2013. Bayes-Ball: The Rational Pastime (for Determining Irrelevance and Requisite Information in Belief Networks and Influence Diagrams). arXiv:1301.7412.

Shipley, B.; and Douma, J. C. 2021. Testing Piecewise Structural Equations Models in the Presence of Latent Variables and Including Correlated Errors. *Structural Equation Modeling: A Multidisciplinary Journal*, 28(4): 582–589.

Spirtes, P.; Glymour, C. N.; and Scheines, R. 2001. *Causation, Prediction, and Search*. MIT Press, 2nd edition. ISBN 9780262194402.

Spirtes, P.; Richardson, T.; Meek, C.; Scheines, R.; and Glymour, C. N. 1998. Using path diagrams as a structural equation modelling tool. *Sociological Methods and Research*, 27(2): 182–225.

Sverchkov, Y.; and Craven, M. 2017. A review of active learning approaches to experimental design for uncovering biological networks. *PLOS Computational Biology*, 13(6): 1–26.

Takata, K. 2010. Space-optimal, backtracking algorithms to list the minimal vertex separators of a graph. *Discrete Applied Mathematics*, 158(15): 1660–1667.

Tennant, P. W. G.; Murray, E. J.; Arnold, K. F.; Berrie, L.; Fox, M. P.; Gadd, S. C.; Harrison, W. J.; Keeble, C.; Ranker, L. R.; Textor, J.; Tomova, G. D.; Gilthorpe, M. S.; and Ellison, G. T. H. 2020. Use of directed acyclic graphs (DAGs) to identify confounders in applied health research: review and recommendations. *International Journal of Epidemiology*, 50(2): 620–632.

Tian, J.; and Pearl, J. 2002a. A General Identification Condition for Causal Effects. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI 2002)*, 567–573. Menlo Park, CA: AAAI Press/The MIT Press.

Tian, J.; and Pearl, J. 2002b. On the Testable Implications of Causal Models with Hidden Variables. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, 519–527. ISBN 1-55860-897-4.

Verma, T. S.; and Pearl, J. 1990. Equivalence and Synthesis of Causal Models. In *Proceedings of the Sixth Conference on*

Uncertainty in Artificial Intelligence, 220–227. Cambridge, MA.

Verma, T. S.; and Pearl, J. 1992. An algorithm for deciding if a set of observed independencies has a causal explanation. In Dubois, D.; Wellman, M.; D'Ambrosio, B.; and Smets, P., eds., *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, 323–330. Stanford, CA: Morgan Kaufmann.

Zantedeschi, V.; Franceschi, L.; Kaddour, J.; Kusner, M. J.; and Niculae, V. 2023. DAG Learning on the Permutahedron. arXiv:2301.11898.

Zhang, J. 2008. Causal Reasoning with Ancestral Graphs. *J. Mach. Learn. Res.*, 9: 1437–1474.

Zheng, Y.; Huang, B.; Chen, W.; Ramsey, J.; Gong, M.; Cai, R.; Shimizu, S.; Spirtes, P.; and Zhang, K. 2024. Causal-learn: Causal discovery in python. *Journal of Machine Learning Research*, 25(60): 1–8.

Appendices

A Background and Previous Work

A.1 Background

In the appendix, for some integer $k \geq 0$, we use $[k]$ to denote the set $\{1, 2, \ldots, k\}$ (with $[0] = \emptyset$).

Graph preliminaries. Let $\mathbf{X}$ be a set of variables in a DAG $\mathcal{G}$ over variables $\mathbf{V}$. We define four kinship relations:

1. *Parents* of $\mathbf{X}$, denoted $Pa(\mathbf{X})$: $Pa(\mathbf{X}) = \{Y \in \mathbf{V} \mid Y \to X \text{ for some } X \in \mathbf{X}\} \cup \mathbf{X}$.
2. *Ancestors* of $\mathbf{X}$, denoted $An(\mathbf{X})$: $An(\mathbf{X}) = \{Y \in \mathbf{V} \mid \text{there is a directed path from } Y \text{ to } X \text{ for some } X \in \mathbf{X}\} \cup \mathbf{X}$.
3. *Descendants* of $\mathbf{X}$, denoted $De(\mathbf{X})$: $De(\mathbf{X}) = \{Y \in \mathbf{V} \mid \text{there is a directed path from } X \text{ to } Y \text{ for some } X \in \mathbf{X}\} \cup \mathbf{X}$
4. *Non-descendants* of $\mathbf{X}$, denoted $Nd(\mathbf{X})$: $\mathbf{V} \setminus De(\mathbf{X})$. Note that $Nd(\mathbf{X})$ does not include $\mathbf{X}$.

Readers may be familiar with spouses of a variable X as variables Y such that X and Y are both the parent of some W. We use a different sense of spouse consistent with (Pearl 2000; Richardson 2003), defined in Section 2.

The ordered local Markov property. We define the ordered local Markov property (Richardson 2003) for semi-Markovian causal DAGs and its basic components below.

Definition A.1.1. (Markov Blanket (MB)) (Richardson 2003) Given a causal graph $\mathcal{G}$ and a consistent ordering $\mathbf{V}^{\prec}$, let X be a variable in $\mathbf{V}^{\prec}$ and $\mathbf{S}$ an ancestral set in $\mathcal{G}$ such that $X \in \mathbf{S} \subseteq \mathbf{V}^{\leq X}$. Then, the Markov blanket of X with respect to the induced subgraph $\mathcal{G}_{\mathbf{S}}$, denoted $mb(X, \mathbf{S})$, is defined as $mb(X, \mathbf{S}) = Pa(\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}}})_{\mathcal{G}_{\mathbf{S}}} \setminus \{X\}$.

Definition A.1.2. (Maximal Ancestral Set (MAS)) (Richardson 2003) Given a causal graph $\mathcal{G}$ and a consistent ordering $\mathbf{V}^{\prec}$, let X be a variable in $\mathbf{V}^{\prec}$ and $\mathbf{S}$ an ancestral set in $\mathcal{G}$ such that $X \in \mathbf{S} \subseteq \mathbf{V}^{\leq X}$. Then, $\mathbf{S}$ is said to be maximal with respect to the Markov blanket $mb(X, \mathbf{S})$ if, for any ancestral set $\mathbf{S}'$ such that $X \in \mathbf{S} \subseteq \mathbf{S}' \subseteq \mathbf{V}^{\leq X}$ and $mb(X, \mathbf{S}) = mb(X, \mathbf{S}')$, we have $\mathbf{S} = \mathbf{S}'$.

We state Richardson's *ordered local Markov property* (with quantification MASs instead of all ancestral sets (Richardson 2003, Section 3.1)).

Definition A.1.3. (The Ordered Local Markov Property (LMP,$\prec$)) (Richardson 2003) A probability distribution $P(\mathbf{v})$ over variables $\mathbf{V}$ is said to satisfy the ordered local Markov property for $\mathcal{G}$ with respect to the consistent ordering $\mathbf{V}^{\prec}$ if, for any variable X and ancestral set $\mathbf{S}$ such that $X \in \mathbf{S} \subseteq \mathbf{V}^{\leq X}$ and $\mathbf{S}$ is maximal with respect to $mb(X, \mathbf{S})$,

$$X \perp\!\!\!\perp \mathbf{S} \setminus (mb(X, \mathbf{S}) \cup \{X\}) \mid mb(X, \mathbf{S}) \text{ in } P(\mathbf{v}).$$

Finally, we introduce the following collections to understand the web of ancestral sets, MBs, and MASs.

Definition A.1.4. Given a causal graph $\mathcal{G}$, a consistent ordering $\mathbf{V}^{\prec}$, and a variable $X \in \mathbf{V}^{\prec}$, define three collections:

$$\mathcal{S}_X = \{X \in \mathbf{S} \subseteq \mathbf{V}^{\leq X} \mid \mathbf{S} \text{ is ancestral }\},$$
$$\mathcal{Z}_X = \{\mathbf{Z} \mid \mathbf{Z} = mb(X, \mathbf{S}) \text{ for some } \mathbf{S} \in \mathcal{S}_X\}, \text{ and}$$
$$\mathcal{S}^+{}_X = \{\mathbf{S}^+ \in \mathcal{S}_X \mid \mathbf{S}^+ \text{ is maximal w.r.t. } mb\big(X, \mathbf{S}^+\big)\}.$$

■ **Example A.1.1.** Consider H in $\mathcal{G}^2$ (Fig. 1b). We have a c-component $\mathcal{C}(7)_{\mathcal{G}^2} = 347$. The ancestral set $\mathbf{S} = 12347$ induces $mb(7, \mathbf{S}) = 234$. The MAS with respect to this MB is $\mathbf{S}^+ = 1234567$, resulting in the CI $7 \perp\!\!\!\perp 156 \mid 234$ invoked by (LMP,$\prec$). □

It is known that GMP and (LMP,$\prec$) are equivalent: for a causal graph $\mathcal{G}$ and consistent ordering $\mathbf{V}^{\prec}$, a probability distribution satisfies the global Markov property for $\mathcal{G}$ if and only if it satisfies the local Markov property for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$ (Richardson 2003, Thm. 2, Section 3.1).

The following lemma provides a (poly-time) test for whether a given set is maximal with respect to the MB that it induces.

Lemma A.1.1. *(Testing Maximality of Ancestral Set) (Richardson 2003, Lemma. 5) Given a causal graph $\mathcal{G}$ and a consistent ordering $\mathbf{V}^{\prec}$, let X be a variable in $\mathbf{V}^{\prec}$. An ancestral set $\mathbf{S} \in \mathcal{S}_X$ is maximal with respect to the Markov blanket $mb(X, \mathbf{S})$ if and only if:*

$$\mathbf{S} = \mathbf{V}^{\leq X} \setminus De(h(X, \mathbf{S}))_{\mathcal{G}}$$

where

$$h(X, \mathbf{S}) = Sp(\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}}})_{\mathcal{G}} \setminus (mb(X, \mathbf{S}) \cup \{X\}).$$

A note on Markov blankets. We offer some clarification on the term 'Markov blanket' as used in this paper (Def. A.1.1), introduced by (Richardson 2003). The more widely known concept of a Markov blanket is due to (Pearl 1988, Def. 3.12). Given a set of variables $\mathbf{V}$ and a variable $X \in \mathbf{V}$, a Pearlian Markov blanket (abbreviated as PMB) is a set of variables $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X\}$ such that $X \perp\!\!\!\perp \mathbf{V} \setminus (\{X\} \cup \mathbf{Z}) \mid \mathbf{Z}$. Returning to Fig. 1b, the variable 8 has a PMB $\mathbf{Z}_1 = 2346$ since $7 \perp\!\!\!\perp 15 \mid 2346$. $\mathbf{Z}_1$ is not an MB (per Def. A.1.1). 8 has another PMB $\mathbf{Z}_2 = 234$ since $7 \perp\!\!\!\perp 156 \mid 234$. $\mathbf{Z}_2$ is also, coincidentally, an MB, though not all MBs are PMBs. The key differences between MBs and PMBs are twofold:

1. In the definition of an MB, we choose an ancestral set $\mathbf{S} \subseteq \mathbf{V}^{\leq X}$ containing X, and require that $X \perp\!\!\!\perp \mathbf{S} \setminus (\{X\} \cup mb(X, \mathbf{S})) \mid mb(X, \mathbf{S})$ holds; the MB separates X from all other variables in $\mathbf{S}$ but not necessarily those in $\mathbf{V} \setminus \mathbf{X}$. The PMB must separate X from all other variables in $\mathbf{V}$.
2. A given ancestral set $\mathbf{S}$ induces exactly one MB for a variable X. However, there may be multiple PMBs for X. An MB is more akin to the notion of a *Markov boundary* (Pearl 1988, Def. 3.12)[5], in the sense that it is 'minimal'; removing any variable Y from the MB $mb(X, \mathbf{S})$ no longer guarantees the independence $X \perp\!\!\!\perp \mathbf{S} \setminus (\{X\} \cup (mb(X, \mathbf{S}) \setminus \{Y\})) \mid mb(X, \mathbf{S}) \setminus \{Y\}$.

MBs, therefore, are closely related to PMBs but with additional features needed to define and ensure that (LMP, $\prec$) is equivalent to GMP.

A.2 Related Work

In this section, we expand on the Markov properties and algorithms to enumerate them summarised in Table A.2.1.

For model testing, a Markov property which invokes only a polynomial number of CI tests is ideal. However, currently known poly-size properties assume either 1) there is no latent confounding between variables, or 2) the given causal DAG does not contain any directed mixed cycles, or 3) the observational distribution satisfies certain additional constraints (Kang and Tian 2009). Intuitively, a directed mixed cycle is a cycle formed by walking through arrows in one direction. For instance, in the causal DAG $\mathcal{G}^2$ (Fig. 1b), the path $2 \to 3 \to 7 \leftrightarrow 3$ is a directed mixed cycle. Directed mixed cycles are commonly found in semi-Markovian DAGs – even in the basic bow pattern, in which a variable X is a cause of Y and X, Y have a latent confounder (Pearl 2000). There is no known poly-sized Markov property for the general setting.

There are two known Markov properties for Markovian causal DAGs.

1. LMP: The local Markov property (Pearl 1988; Lauritzen et al. 1990; Lauritzen 1996). LMP specifies a linear number of CIs in total: one for each variable X, stating that X is conditionally independent of its non-descendants given its parents.
2. PMP: The pairwise Markov property (Pearl and Meshkat 1999). For a graph with n variables, PMP invokes $O(n^2)$ CIs: more specifically, one CI for each pair of non-adjacent variables. PMP assumes that the given probability distribution is a compositional graphoid: that is, it additionally satisfies the intersection and composition axioms.

The intersection and composition axioms do not hold in arbitrary distributions. The intersection axioms holds, for example, in distributions which have full support ($P(\mathbf{v}) > 0$ for all $\mathbf{v}$), e.g., a multivariate Gaussian. Composition holds in multivariate Gaussians and in probability distributions that are faithful to some DAG.

The following are known Markov properties for semi-Markovian causal DAGs.

1. RLMP: The reduced local Markov property (Kang and Tian 2009). RLMP invokes a linear number of CIs in total, one for each variable. RLMP states that a variable is independent of the variables that are neither its descendants nor the descendants of its spouses, conditioning on its parents. The property assumes that the given probability distribution satisfies the composition axiom and the DAG has no directed mixed cycles.
2. (RLMP,$\prec$): The ordered reduced local Markov property (Kang and Tian 2009). Given a specific ordering of variables called a *c-ordering* (Kang and Tian 2009), (RLMP,$\prec$) invokes a linear number of CIs in total. (RLMP,$\prec$) states that each variable is independent of its predecessors (excluding its spouses) in a c-ordering, given its parents. The property assumes that the given probability distribution satisfies the composition axiom and the DAG has no directed mixed cycles.
3. PMP-C: The pairwise Markov property (Kang and Tian 2009). Given a c-ordering, PMP-C invokes $O(n^2)$ many CIs: more specifically, one CI for each pair of non-adjacent variables. PMP-C assumes that the given probability distribution satisfies the composition axiom and the DAG has no directed mixed cycles.
4. PMP-RS: The pairwise Markov property given by (Richardson and Spirtes 2002). PMP-RS invokes $O(n^2)$

[5] A Markov boundary is a minimal (Pearlian) Markov blanket, such that any strict subset of the Markov boundary no longer separates the variable from all other variables in the graph.

Property	Coverage		Scalability	
	Latents	**Any Prob. Distr.**	**Poly-size CIs**	**Poly-Delay**
LMP	✗	✓	✓	✓
PMP	✗	▲	✓	✓
RLMP	▲	▲	✓	✓
(RLMP,$\prec$)	▲	▲	✓	✓
PMP-C	▲	▲	✓	✓
PMP-RS	▲	▲	✓	✓
S-Markov	✓	✓	✗	✗
(LMP,$\prec$)	✓	✓	✗	✗
C-LMP (ours)	✓	✓	✗	✓

Table A.2.1: Summary of properties and algorithms to enumerate CIs invoked by such properties. The first column denotes if the property applies to graphs with unobserved confounders; the second, if it applies to arbitrary observational distributions; the third, if it invokes a polynomial number of CIs; the fourth, if there is a poly-delay algorithm to list its invoked CIs. ✓ denotes an addressed area. ✗ denotes an unaddressed area. ▲denotes that DAGs may contain unobserved variables but not directed mixed cycles (Kang and Tian 2009), or the input is a MAG, a tranformation of a DAG (Richardson and Spirtes 2002). ▲denotes that further assumptions must be made on the probability distribution.

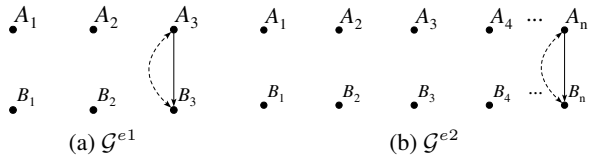
many CIs, one for each pair of non-adjacent variables, for a given *maximal ancestral graph* (MAG). A semi-Markovian DAG can be transformed into a MAG which encodes exactly the same CIs. It thus suffices to test CIs in the resultant MAG (Shipley and Douma 2021). However, the equivalence between this pairwise Markov property and the global Markov property has only been proved for probability distributions that are compositional graphoids (Lauritzen and Sadeghi 2018).

5. S-Markov: The S-Markov property (Kang and Tian 2009). S-Markov relaxes the assumption of the given graph containing no directed mixed cycles. Still, S-Markov assumes that the observational distribution satisfies the composition axiom. For each variables in the graph that can be c-ordered, S-Markov invokes a linear number of CIs. However, for variables that are not c-ordered, S-Markov relies on the ordered local Markov property (LMP,$\prec$), which, as discussed, is exponential-sized.

CIs are the only type of constraint that Markovian DAGs impose on the observational distribution. In the non-Markovian case, however, DAGs may encode more complex equality and inequality constraints such as *Verma constraints* (Verma and Pearl 1990). While such constraints are outside the scope of this work, there are algorithms that list these constraints in addition to CIs. However, these algorithms do not run in poly-delay.

Algorithm B.1.1: ListCIBF $(\mathcal{G}, \mathbf{V}^{\prec})$

Input: $\mathcal{G}$ a causal diagram; $\mathbf{V}^{\prec}$ an ordering consistent with $\mathcal{G}$.
Output: Listing CIs invoked by (LMP,$\prec$) for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$.
for each $X \in \mathbf{V}^{\prec}$ **do**
  for each ancestral set $\mathbf{S}$ such that $X \in \mathbf{S} \subseteq \mathbf{V}^{\leq X}$ **do**
    if $\mathbf{S}$ is maximal with respect to $mb(X, \mathbf{S})$[6] **then**
      Output $X \perp\!\!\!\perp \mathbf{S} \setminus (mb(X, \mathbf{S}) \cup \{X\}) \mid mb(X, \mathbf{S})$



(a) $\mathcal{G}^{e1}$ (b) $\mathcal{G}^{e2}$

Figure B.1.1: Examples showing that a brute-force approach (ListCIBF) may take exponential time to output one CI invoked by (LMP,$\prec$).

B C-LMP and the Ordered Local Markov Property

(LMP, $\prec$) is a well-known Markov property that applies to arbitrary observational distributions and causal graphs with unobserved confounders. In this section, we first explain how naively following the definition of (LMP,$\prec$) can take exponential time to output just one CI. Next, we characterize (LMP, $\prec$) in more depth and show how ACs (Def. 4) can be used to compute the CIs that (LMP, $\prec$) invokes.

B.1 Brute-Force Listing of CIs Invoked by (LMP,$\prec$)

By definition, we can list the CIs invoked by (LMP,$\prec$) (Def. A.1.3) by enumerating over MASs. However, it is unclear how to enumerate over MASs. Each MAS is defined relative to an MB, and each MB is defined relative to an ancestral set. Then, an immediate approach is to iterate over all ancestral sets $\mathbf{S}$, verifying if $\mathbf{S}$ is maximal with respect to $mb(X, \mathbf{S})$ before we output its corresponding CI constraint. We implement this approach in the algorithm ListCIBF (Alg. B.1.1).

■ **Example B.1.1.** Consider the DAG $\mathcal{G}^{e1}$ (Fig. B.1.1a) with consistent ordering $\mathbf{V}^{\prec} = \{A_1, A_2, A_3, B_1, B_2, B_3\}$. ListCIBF$(\mathcal{G}^{e1}, \mathbf{V}^{\prec})$ outputs five CIs invoked by (LMP,$\prec$): $A_2 \perp\!\!\!\perp \{A_1\}$, $A_3 \perp\!\!\!\perp \{A_1, A_2\}$, $B_1 \perp\!\!\!\perp \{A_1, A_2, A_3\}$, $B_2 \perp\!\!\!\perp \{A_1, A_2, A_3, B_2\}$, and $B_3 \perp\!\!\!\perp \{A_1, A_2, B_1, B_2\} \mid \{A_3\}$. □

In Ex. B.1.1, given $X = B_3$, ListCIBF$(\mathcal{G}^{e1}, \mathbf{V}^{\prec})$ iterates over 2^4 different ancestral sets $\mathbf{S}$ with $B_3 \in \mathbf{S} \subseteq \mathbf{V}^{\leq B_3}$, all of which produce the same $mb(B_3, \mathbf{S}) = \{A_3\}$. However, only $\mathbf{S}^{+} = \mathbf{V}^{\leq B_3}$ is maximal with respect to this MB, resulting in the CI: $B_3 \perp\!\!\!\perp \{A_1, A_2, B_1, B_2\} \mid \{A_3\}$. ListCIBF goes over 2^4 different ancestral sets to output this CI. Next, we generalize this example to show that ListCIBF may iterate over exponentially many ancestral sets (with respect to

the number of variables in $\mathcal{G}$) that produce the same MB.

■ **Example B.1.2.** In $\mathcal{G}^{e2}$ (Fig. B.1.1b) with $2n$ nodes, there are $2^{n-1}+2^{n-2}-1$ ancestral sets and n of them are maximal. □

In other words, iterating over all ancestral sets naively is potentially sub-optimal.

In the following lemma, we make a key observation: while there may be many ancestral sets producing the same MB (so that $|\mathcal{S}_X| > |\mathcal{Z}_X|$), exactly one ancestral set is maximal with respect to this MB. As a result, LISTCIBF may take exponential time to output just one new CI.

Lemma B.1.1 (One-to-one Correspondence between $\mathcal{Z}_X$ and ${\mathcal{S}^+}_X$)**.** *Given a causal graph $\mathcal{G}$ and a consistent ordering $\mathbf{V}^{\prec}$, let X be a variable in $\mathbf{V}^{\prec}$. There is a bijection $f : \mathcal{Z}_X \to {\mathcal{S}^+}_X$ given by $f(\mathbf{Z}) = \mathbf{S}^+$ where $\mathbf{S}^+ \in {\mathcal{S}^+}_X$ is an ancestral set maximal with respect to $\mathbf{Z} \in \mathcal{Z}_X$. The inverse of f, $g : {\mathcal{S}^+}_X \to \mathcal{Z}_X$, is given by $g(\mathbf{S}^+) = mb(X, \mathbf{S}^+)$.*

■ **Example B.1.3.** Continuing Ex. B.1.1. Given a variable B_3, there exists only one MAS $\mathbf{V}^{\leq B_3}$ with respect to the MB $\mathbf{Z} = \{A_3\}$ of B_3. We have $\mathcal{Z}_{B_3} = \{\{A_3\}\}$ and ${\mathcal{S}^+}_{B_3} = \{\mathbf{V}^{\leq B_3}\}$. $\mathbf{V}^{\leq B_3}$ maps uniquely to $\{A_3\}$, and vice versa. □

B.2 Computing MBs and MASs using ACs

Listing CIs invoked by (LMP, $\prec$) is challenging due to the many-to-one mapping from ancestral sets to CIs. Minimally, we want to be able to list these CIs without brute-force iteration. Fundamental to our solution is the fact that multiple ancestral sets induce the same CI only because they induce the same AC (Def. 4).

Observe that exponentially many ancestral sets may induce the same AC. For instance, in $\mathcal{G}^{e1}$ (Fig. B.1.1a) with $X = B_3$, 2^4 different ancestral sets $\mathbf{S} \in \mathcal{S}_{B_3}$ induce the same AC, $\mathcal{C}(B_3)_{\mathcal{G}_\mathbf{S}} = \{A_3, B_3\}$.

We show that all ancestral sets inducing the same MB and MAS must induce the same AC.

Proposition B.2.1 (Equality of MBs Implies Equality of ACs)**.** *Given a causal graph $\mathcal{G}$ and a consistent ordering $\mathbf{V}^{\prec}$, for any variable $X \in \mathbf{V}^{\prec}$ and any ancestral sets $\mathbf{S}_1, \mathbf{S}_2 \in \mathcal{S}_X$, if $mb(X, \mathbf{S}_1) = mb(X, \mathbf{S}_2)$, then $\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}} = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_2}}$.*

Moreover, the converse is also true: all ancestral sets inducing the same AC must induce the same MB and MAS. In particular, for a variable X, given $\mathbf{C} = \mathcal{C}(X)_{\mathcal{G}_\mathbf{S}}$ for some ancestral set $\mathbf{S}$, we can compute $\mathbf{Z} = mb(X, \mathbf{S})$ and the MAS $\mathbf{S}^+$ relative to $\mathbf{Z}$ in poly-time without using $\mathbf{S}$. The following results show how MB and MAS can be computed from AC.

Proposition B.2.2 (Construction of MB from AC)**.** *Given a causal graph $\mathcal{G}$ and a consistent ordering $\mathbf{V}^{\prec}$, let X be a variable in $\mathbf{V}^{\prec}$. Fix an ancestral c-component $\mathbf{C} \in {\mathcal{AC}}_X$. For any ancestral set $\mathbf{S} \in \mathcal{S}_X$ such that $\mathcal{C}(X)_{\mathcal{G}_\mathbf{S}} = \mathbf{C}$, we have $mb(X, \mathbf{S}) = Pa(\mathbf{C}) \setminus \{X\}$.*

Proposition B.2.3 (Construction of MAS from AC)**.** *Given a causal graph $\mathcal{G}$ and a consistent ordering $\mathbf{V}^{\prec}$, let X be a variable $\mathbf{V}^{\prec}$. Fix an ancestral c-component $\mathbf{C} \in {\mathcal{AC}}_X$. For any ancestral set $\mathbf{S} \in \mathcal{S}_X$ such that $\mathcal{C}(X)_{\mathcal{G}_\mathbf{S}} = \mathbf{C}$, the unique ancestral set $\mathbf{S}^+ \in {\mathcal{S}^+}_X$ maximal with respect to the Markov blanket $mb(X, \mathbf{S})$ is given by $\mathbf{S}^+ = \mathbf{V}^{\leq X} \setminus De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C}))$.*

■ **Example B.2.1.** Consider the DAG $\mathcal{G}^{e1}$ (Fig. B.1.1a) with consistent ordering $\mathbf{V}^{\prec} = \{A_1, A_2, A_3, B_1, B_2, B_3\}$. Given a variable B_3, $\mathbf{C} = \{A_3, B_3\}$ is an AC relative to B_3. We compute the MB $\mathbf{Z}$ from $\mathbf{C}$ as follows: $Pa(\{A_3, B_3\}) \setminus \{B_3\} = \{A_3\} = \mathbf{Z}$. For all ancestral sets $\mathbf{S} \in \mathcal{S}_{B_3}$, we have $mb(B_3, \mathbf{S}) = Pa(\mathcal{C}(B_3)_{\mathcal{G}_\mathbf{S}})_{\mathcal{G}_\mathbf{S}} \setminus \{B_3\} = \{A_3\} = \mathbf{Z}$. The MAS $\mathbf{S}^+$ relative to $\mathbf{Z}$ is given by $\mathbf{S}^+ = \mathbf{V}^{\leq B_3} \setminus De(Sp(\{A_3, B_3\}) \setminus Pa(\{A_3, B_3\})) = \mathbf{V}^{\leq B_3} \setminus De(\emptyset) = \mathbf{V}^{\leq B_3} = \mathbf{S}^+$. □

These results, in part, motivate our definition of a local Markov property via ancestral c-components i.e., C-LMP (Def. 5). In fact, we can show the following equivalence between C-LMP and (LMP,$\prec$).

Theorem B.2.1 (Correspondence between C-LMP and (LMP,$\prec$))**.** *Let $\mathcal{G}$ be a causal graph and $\mathbf{V}^{\prec}$ a consistent ordering. The c-component local Markov property and the ordered local Markov property (Richardson 2003) for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$ induce an identical set of conditional independence relations implied by $\mathcal{G}$ over $\mathbf{V}$.*

Proof. Given a causal graph $\mathcal{G}$ and a consistent ordering $\mathbf{V}^{\prec}$, let $\mathcal{L}^R$ denote the set of CIs implied by the ordered local Markov property for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$, and $\mathcal{L}^C$ the set of CIs implied by the c-component local Markov property for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$. We show that $\mathcal{L}^R = \mathcal{L}^C$.

1. ($\mathcal{L}^R \subseteq \mathcal{L}^C$) Consider a CI statement in $\mathcal{L}^R$ of the form
$$X \perp\!\!\!\perp \mathbf{S}^+ \setminus (mb(X, \mathbf{S}^+) \cup \{X\}) \mid mb(X, \mathbf{S}^+)$$
for some variable $X \in \mathbf{V}^{\prec}$ and an ancestral set $\mathbf{S}^+ \in {\mathcal{S}^+}_X$ maximal with respect to $mb(X, \mathbf{S}^+)$. We show that the same CI statement is also in $\mathcal{L}^C$.
Let $\mathbf{C} = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}^+}}$. Since $\mathbf{S}^+$ is ancestral, $\mathbf{C}$ is an AC relative to X. By Def. 5, the following CI is in $\mathcal{L}^C$.
$$X \perp\!\!\!\perp \mathbf{S}^{+'} \setminus Pa(\mathbf{C}) \mid (Pa(\mathbf{C}) \setminus \{X\})$$
where
$$\mathbf{S}^{+'} = \mathbf{V}^{\leq X} \setminus De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C}))$$
By Prop. B.2.2, $mb(X, \mathbf{S}^+) = Pa(\mathbf{C}) \setminus \{X\}$. By Prop. B.2.3, $\mathbf{S}^+ = \mathbf{S}^{+'}$. Therefore, the two CI statements are identical, and the given CI from $\mathcal{L}^R$ is also in $\mathcal{L}^C$.
2. ($\mathcal{L}^C \subseteq \mathcal{L}^R$) Consider a CI statement in $\mathcal{L}^C$ of the form
$$X \perp\!\!\!\perp \mathbf{S}^+ \setminus Pa(\mathbf{C}) \mid Pa(\mathbf{C}) \setminus \{X\}$$
where
$$\mathbf{S}^+ = \mathbf{V}^{\leq X} \setminus De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C}))$$
for some variable $X \in \mathbf{V}^{\prec}$ and AC $\mathbf{C} \in {\mathcal{AC}}_X$. By Def. 4, there exists an ancestral set $\mathbf{S} \in \mathcal{S}_X$ such that $\mathbf{C} = \mathcal{C}(X)_{\mathcal{G}_\mathbf{S}}$. By Prop. B.2.2, $mb(X, \mathbf{S}) = Pa(\mathbf{C}) \setminus \{X\}$. By Prop. B.2.3, $\mathbf{S}^+$ is the unique ancestral set maximal with

[6] A poly-time test for whether an ancestral set $\mathbf{S}$ is maximal with respect to $mb(X, \mathbf{S})$ is shown in Lemma A.1.1.

respect to $mb(X, \mathbf{S})$. By Def. A.1.3, the following CI is in $\mathcal{L}^R$

$$X \perp\!\!\!\perp \mathbf{S}^+ \setminus (mb(X, \mathbf{S}^+) \cup \{X\}) \mid mb(X, \mathbf{S}^+)$$

Therefore, the two CI statements are identical, and the given CI from $\mathcal{L}^C$ is also in $\mathcal{L}^R$.

□

The equivalence between C-LMP and GMP (Thm. 1) can also be proved as a corollary of the above Thm. B.2.1 and the equivalence between (LMP,$\prec$) and GMP (Richardson 2003, Thm. 2, Section 3.1).

B.3 Uniqueness Property of ACs

Recall that in (LMP, $\prec$), multiple ancestral sets can induce the same MB. Here, we show this can be remedied using ACs: each MB can be computed from exactly one AC.

Lemma B.3.1 (One-to-one Correspondence between $\mathcal{AC}_X$ and $\mathcal{Z}_X$)**.** *Let $\mathcal{G}$ be a causal graph, $\mathbf{V}^{\prec}$ a consistent ordering, and X a variable in $\mathbf{V}^{\prec}$. Then, there is a bijection $f : \mathcal{AC}_X \to \mathcal{Z}_X$ given by $f(\mathbf{C}) = Pa(\mathbf{C}) \setminus \{X\}$ with $\mathbf{C} \in \mathcal{AC}_X$. The inverse of f, $g : \mathcal{Z}_X \to \mathcal{AC}_X$, is given by $g(\mathbf{Z}) = \mathcal{C}(X)_{\mathcal{G}_\mathbf{S}}$ where $\mathbf{S}$ is an arbitrary ancestral set in $\mathcal{S}_X$ such that $\mathbf{Z} = mb(X, \mathbf{S})$.*

Both Lemma B.1.1 and Lemma B.3.1 imply a one-to-one correspondence between ACs and MASs.

Corollary B.3.1 (One-to-one Correspondence between $\mathcal{AC}_X$ and $\mathcal{S}^+{}_X$)**.** *Let $\mathcal{G}$ be a causal graph, $\mathbf{V}^{\prec}$ a consistent ordering, and X a variable in $\mathbf{V}^{\prec}$. There is a bijection $f : \mathcal{AC}_X \to \mathcal{S}^+{}_X$.*

Fig. B.4.1 provides an overview of the relationships among the sets of ancestral sets, ACs, MBs, and MASs. A core implication is that each CI invoked by (LMP,$\prec$) can be derived from exactly one AC, which we exploit in C-LMP (Def. 5).

B.4 Proofs

We present proofs of the results in Sections B.2 and B.3. We first prove some technical propositions.

Proposition B.4.1 (AC in Union of Subgraphs)**.** *Given a causal graph $\mathcal{G}$ over a set of variables $\mathbf{V}$, for any subsets $\mathbf{S}_1, \mathbf{S}_2 \subseteq \mathbf{V}$ and a variable $X \in \mathbf{V}$, if $\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}} = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_2}} = \mathbf{C}$ then $\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1 \cup \mathbf{S}_2}} = \mathbf{C}$.*

Proof. Since $\mathbf{S}_1 \subseteq \mathbf{S}_1 \cup \mathbf{S}_2$, we have $\mathbf{C} \subseteq \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1 \cup \mathbf{S}_2}}$. To show the other direction, for any variable $U \in \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1 \cup \mathbf{S}_2}} \setminus \{X\}$, let $\pi = \{X \leftrightarrow V_1, V_1 \leftrightarrow V_2, \ldots, V_{k-1} \leftrightarrow V_k, V_k \leftrightarrow V_{k+1} = U\}$ be the bidirected path from X to U in $\mathcal{G}_{\mathbf{S}_1 \cup \mathbf{S}_2}$ (for some $k \geq 0$). For each $i \in [k+1]$, we have $V_i \in \mathbf{S}_1 \cup \mathbf{S}_2$. We prove by induction on the index $i \in [k+1]$ that $V_i \in \mathbf{C}$ for each $i \in [k+1]$.

Base case. If $V_1 \in \mathbf{S}_1$then $X \leftrightarrow V_1$ implies $V_1 \in C(X)_{\mathcal{G}_{\mathbf{S}_1}} = \mathbf{C}$. Otherwise, $V_1 \in \mathbf{S}_2$ implies $V_1 \in C(X)_{\mathcal{G}_{\mathbf{S}_2}} = \mathbf{C}$.

Inductive hypothesis. If $k \geq 1$, assume for some $i \in [k]$ we have $V_i \in \mathbf{C} = C(X)_{\mathcal{G}_{\mathbf{S}_1}} = C(X)_{\mathcal{G}_{\mathbf{S}_2}}$.

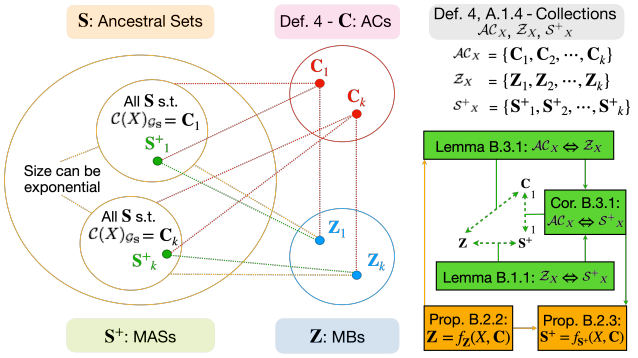


Figure B.4.1: An overview of the relationships among ancestral sets, ACs, MBs, MASs. Exponentially many ancestral sets may map to one AC, MB, or MAS. On the other hand, there is a one-to-one-correspondence among ACs, MBs and MASs.

Inductive step. Then, either $V_{i+1} \in \mathbf{S}_1$ or $V_{i+1} \in \mathbf{S}_2$. If $V_{i+1} \in \mathbf{S}_1$, then $V_i \in C(X)_{\mathcal{G}_{\mathbf{S}_1}}$ (by the induction hypothesis) and $V_i \leftrightarrow V_{i+1}$ implies $V_{i+1} \in C(X)_{\mathcal{G}_{\mathbf{S}_1}}$. Otherwise, $V_{i+1} \in \mathbf{S}_2$ and $V_i \in C(X)_{\mathcal{G}_{\mathbf{S}_2}}$ (by the induction hypothesis) $V_i \leftrightarrow V_{i+1}$ implies $V_{i+1} \in C(X)_{\mathcal{G}_{\mathbf{S}_2}}$. By induction, it follows that $U = V_{k+1} \in \mathbf{C}$. □

Proof of Prop. B.2.1. Consider $\mathbf{S}_1, \mathbf{S}_2 \in \mathcal{S}_X$. If $mb(X, \mathbf{S}_1) = mb(X, \mathbf{S}_2)$, then $mb(X, \mathbf{S}_1) \subseteq \mathbf{S}_2$ and $mb(X, \mathbf{S}_2) \subseteq \mathbf{S}_1$ since $mb(X, \mathbf{S}_1) \subseteq \mathbf{S}_1$, $mb(X, \mathbf{S}_2) \subseteq \mathbf{S}_2$. This implies $\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}} \subseteq mb(X, \mathbf{S}_1) \cup \{X\} \subseteq \mathbf{S}_2$ and $\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_2}} \subseteq mb(X, \mathbf{S}_2) \cup \{X\} \subseteq \mathbf{S}_1$. However, $\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}} \subseteq \mathbf{S}_2 \implies \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}} \subseteq \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_2}}$ and similarly, $\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_2}} \subseteq \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}}$. Therefore, $\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}} = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_S}}$. □

Proposition B.4.2 (MB in Union of Subgraphs)**.** *Given a causal graph $\mathcal{G}$ and a consistent ordering $\mathbf{V}^{\prec}$, for any variable $X \in \mathbf{V}$ and any ancestral sets $\mathbf{S}_1, \mathbf{S}_2 \in \mathcal{S}_X$, if $mb(X, \mathbf{S}_1) = mb(X, \mathbf{S}_2) = \mathbf{Z}$, then $mb(X, \mathbf{S}_1 \cup \mathbf{S}_2) = \mathbf{Z}$.*

Proof. Consider $\mathbf{S}_1, \mathbf{S}_2 \in \mathcal{S}_X$. If $mb(X, \mathbf{S}_1) = mb(X, \mathbf{S}_2)$, then by Prop. B.2.1, we have $\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}} = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_2}}$. Since $\mathbf{S}_1, \mathbf{S}_2$ are ancestral, we have $An(\mathbf{S}_1 \cup \mathbf{S}_2)_\mathcal{G} = An(\mathbf{S}_1)_\mathcal{G} \cup An(\mathbf{S}_2)_\mathcal{G} = \mathbf{S}_1 \cup \mathbf{S}_2$, hence $\mathbf{S}_1 \cup \mathbf{S}_2$ is also ancestral. We get

$$\begin{aligned} mb(X, \mathbf{S}_1 \cup \mathbf{S}_2) &= Pa\big(\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1 \cup \mathbf{S}_2}}\big)_{\mathcal{G}_{\mathbf{S}_1 \cup \mathbf{S}_2}} \setminus \{X\} \quad (5) \\ &= Pa\big(\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}}\big)_{\mathcal{G}_{\mathbf{S}_1 \cup \mathbf{S}_2}} \setminus \{X\} \\ &\qquad \text{(By Prop. B.4.1 since } \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}} = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_2}}) \\ &= Pa\big(\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}}\big)_{\mathcal{G}_{\mathbf{V}^{\leq X}}} \setminus \{X\} \\ &\qquad (\mathbf{S}_1 \cup \mathbf{S}_2 \text{ ancestral}) \\ &= Pa\big(\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}}\big)_{\mathcal{G}_{\mathbf{S}_1}} \setminus \{X\} \\ &\qquad (\mathbf{S}_1 \text{ ancestral}) \\ &= mb(X, \mathbf{S}_1) \quad (6) \end{aligned}$$

□

We now move to our main results in Sections B.2. and B.3.

Proof of Lemma B.1.1. First, we show the mapping f is well-defined. Given $\mathbf{Z} \in \mathcal{Z}_X$, there exists an ancestral set $\mathbf{S}^+ \in \mathcal{S}^+{}_X$ maximal with respect to $\mathbf{Z}$. It remains to show that there is exactly one such $\mathbf{S}^+$. Let $\mathbf{S}_1, \mathbf{S}_2 \in \mathcal{S}^+{}_X$ be ancestral sets maximal with respect to $\mathbf{Z}$. The equality $mb(X, \mathbf{S}_1) = mb(X, \mathbf{S}_2) = \mathbf{Z}$ implies $mb(X, \mathbf{S}_1 \cup \mathbf{S}_2) = \mathbf{Z}$ (by Prop. B.4.2). Therefore, $\mathbf{S}_1 \subseteq \mathbf{S}_1 \cup \mathbf{S}_2$ and the maximality of $\mathbf{S}_1$ implies $\mathbf{S}_1 = \mathbf{S}_1 \cup \mathbf{S}_2$ and $\mathbf{S}_2 \subseteq \mathbf{S}_1$. Similarly, $\mathbf{S}_1 \subseteq \mathbf{S}_2$. Therefore, $\mathbf{S}_1 = \mathbf{S}_2$.

Finally, g is well-defined. $f(g(\mathbf{S}^+)) = f(mb(X, \mathbf{S}^+)) = \mathbf{S}^+$, and $g(f(\mathbf{Z})) = \mathbf{Z}$ since $f(\mathbf{Z}) = \mathbf{S}^+$ is maximal with respect to $\mathbf{Z}$ if and only if $mb(X, \mathbf{S}^+) = \mathbf{Z}$. Since f has a two-sided inverse g, f is bijective. □

Proof of Prop. B.2.2. For any ancestral set $\mathbf{S} \in \mathcal{S}_X$ with $\mathcal{C}(X)_{\mathcal{G}_\mathbf{S}} = \mathbf{C}$, we have

$$\begin{aligned} mb(X, \mathbf{S}) &= Pa(\mathcal{C}(X)_{\mathcal{G}_\mathbf{S}})_{\mathcal{G}_\mathbf{S}} \backslash \{X\} && (7) \\ &= Pa(\mathbf{C})_{\mathcal{G}_\mathbf{S}} \backslash \{X\} && \text{(By definition of } \mathbf{S}) \\ &= Pa(\mathbf{C}) \backslash \{X\} && (\mathbf{S} \text{ is ancestral and } \mathbf{C} \subseteq \mathbf{S}) \end{aligned}$$

□

Proof of Prop. B.2.3. For any ancestral set $\mathbf{S} \in \mathcal{S}_X$ with $\mathcal{C}(X)_{\mathcal{G}_\mathbf{S}} = \mathbf{C}$, let $\mathbf{S}^+ \in \mathcal{S}^+{}_X$ be an ancestral set maximal with respect to $mb(X, \mathbf{S})$. Note that $mb(X, \mathbf{S}) = mb(X, \mathbf{S}^+)$ by definition. By Prop. B.2.1, we have $\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}^+}} = \mathcal{C}(X)_{\mathcal{G}_\mathbf{S}} = \mathbf{C}$. Then, $\mathbf{S}^+$ is maximal with respect to $mb(X, \mathbf{S}^+)$ if and only if

$$\begin{aligned} \mathbf{S}^+ &= \mathbf{V}^{\leq X} \setminus De\big(Sp\big(\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}^+}}\big) \setminus \big(mb\big(X, \mathbf{S}^+\big) \cup \{X\}\big)\big) \\ &\qquad \text{(Lemma A.1.1)} \\ &= \mathbf{V}^{\leq X} \setminus De\big(Sp(\mathbf{C}) \setminus \big(mb\big(X, \mathbf{S}^+\big) \cup \{X\}\big)\big) && (8) \\ &= \mathbf{V}^{\leq X} \setminus De(Sp(\mathbf{C}) \setminus ((Pa(\mathbf{C}) \setminus \{X\}) \cup \{X\})) \\ &\qquad \text{(Prop. B.2.2)} \\ &= \mathbf{V}^{\leq X} \setminus De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})). && (9) \end{aligned}$$

The uniqueness of $\mathbf{S}^+$ follows from Lemma B.1.1. Note that $\mathbf{S}^+$ depends only on $\mathbf{C}$, not the particular $\mathbf{S}$ such that $\mathcal{C}(X)_{\mathcal{G}_\mathbf{S}} = \mathbf{C}$. □

Proof of Lemma B.3.1. First, we show the mapping f is well-defined. Given $\mathbf{C} \in \mathcal{AC}_X$, by definition, there exists an ancestral set $\mathbf{S} \in \mathcal{S}_X$ such that $\mathbf{C} = \mathcal{C}(X)_{\mathcal{G}_\mathbf{S}}$. Then, $f(\mathbf{C}) = Pa(\mathbf{C}) \setminus \{X\} = mb(X, \mathbf{S})$ by Prop. B.2.2, so $f(\mathbf{C}) \in \mathcal{Z}_X$ holds.

Next, we show that f is bijective by exhibiting an inverse $g : \mathcal{Z}_X \to \mathcal{AC}_X$. Given $\mathbf{Z} \in \mathcal{Z}_X$, fix any $\mathbf{S} \in \mathcal{S}_X$ such that $\mathbf{Z} = mb(X, \mathbf{S})$ (we know such $\mathbf{S}$ exists by the definition of $\mathbf{Z}$) and let $g(\mathbf{Z}) = \mathcal{C}(X)_{\mathcal{G}_\mathbf{S}}$.

To see that g is well-defined, first note that $\mathcal{C}(X)_{\mathcal{G}_\mathbf{S}} \in \mathcal{AC}_X$ since $\mathbf{S}$ is an ancestral set by assumption. Second, we need to show that $g(\mathbf{Z})$ is independent of the particular choice of $\mathbf{S}$ (since multiple ancestral sets can induce the same MB). Consider $\mathbf{S}_1, \mathbf{S}_2 \in \mathcal{S}_X$ such that $\mathbf{Z} = mb(X, \mathbf{S}_1) = mb(X, \mathbf{S}_2)$. Let $\mathbf{C}_1 = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}}$ and $\mathbf{C}_2 = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_2}}$.

We show that $\mathbf{C}_1 = \mathbf{C}_2$. By Prop. B.2.2, $mb(X, \mathbf{S}_1) = Pa(\mathbf{C}_1) \setminus \{X\}$ and $mb(X, \mathbf{S}_2) = Pa(\mathbf{C}_2) \setminus \{X\}$. From the equality $mb(X, \mathbf{S}_1) = mb(X, \mathbf{S}_2)$, we have $Pa(\mathbf{C}_1) \setminus \{X\} = Pa(\mathbf{C}_2) \setminus \{X\}$ and hence $Pa(\mathbf{C}_1) = Pa(\mathbf{C}_2)$. Since $\mathbf{S}_1, \mathbf{S}_2$ are ancestral, we have $Pa(\mathbf{C}_1) \subseteq \mathbf{S}_1$, and $Pa(\mathbf{C}_2) \subseteq \mathbf{S}_2$. With $Pa(\mathbf{C}_1) \setminus \{X\} = Pa(\mathbf{C}_2) \setminus \{X\}$, we have $\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}} = \mathbf{C}_1 \subseteq Pa(\mathbf{C}_1) = Pa(\mathbf{C}_2) \subseteq \mathbf{S}_2$, implying $\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}} \subseteq \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_2}}$. By a symmetric argument, we get $\mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_2}} \subseteq \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}_1}}$. Therefore, $\mathbf{C}_1 = \mathbf{C}_2$.

Finally, we show that g is a two-sided inverse of f. Given $\mathbf{C} \in \mathcal{AC}_X$, fix some $\mathbf{S} \in \mathcal{S}_X$ such that $\mathbf{C} = \mathcal{C}(X)_{\mathcal{G}_\mathbf{S}}$. Then

$$\begin{aligned} g(f(\mathbf{C})) &= g(Pa(\mathbf{C}) \setminus \{X\}) && (10) \\ &= g(mb(X, \mathbf{S})) && \text{(Prop. B.2.2)} \\ &= \mathcal{C}(X)_{\mathcal{G}_\mathbf{S}} && \text{(By definition of } g) \\ &= \mathbf{C} && (11) \end{aligned}$$

Given $\mathbf{Z} \in \mathcal{Z}_X$, fix some $\mathbf{S} \in \mathcal{S}_X$ such that $\mathbf{Z} = mb(X, \mathbf{S})$. Then,

$$\begin{aligned} f(g(\mathbf{Z})) &= f(\mathcal{C}(X)_{\mathcal{G}_\mathbf{S}}) && (12) \\ &= Pa(\mathcal{C}(X)_{\mathcal{G}_\mathbf{S}}) \setminus \{X\} && (13) \\ &= Pa(\mathcal{C}(X)_{\mathcal{G}_\mathbf{S}})_{\mathcal{G}_\mathbf{S}} \setminus \{X\} \\ &\qquad (\mathbf{S} \text{ is ancestral and } \mathcal{C}(X)_{\mathcal{G}_\mathbf{S}} \subseteq \mathbf{S}) \\ &= mb(X, \mathbf{S}) && (14) \\ &= \mathbf{Z} && (15) \end{aligned}$$

□

Proof of Cor. B.3.1. The result follows from Lemma B.1.1 and Lemma B.3.1, composing the bijective mappings $f_1 : \mathcal{AC}_X \to \mathcal{Z}_X$ and $f_2 : \mathcal{Z}_X \to \mathcal{S}^+{}_X$. □

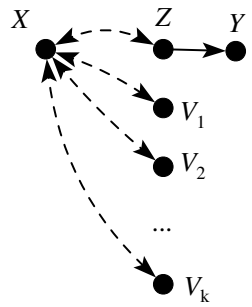


Figure B.4.2: A causal graph $\mathcal{G}$ with consistent ordering $Z \prec Y \prec X \prec V_1 \prec \cdots \prec V_k$, inducing $\Omega(2^n)$ number of CIs invoked by C-LMP.

C Proofs

C.1 Section 3 Proofs

Proposition C.1.1. *Let $\mathcal{G}$ be a causal graph and $\mathbf{V}^\prec$ a consistent ordering. For any variable $X \in \mathbf{V}^\prec$ and any ancestral c-component $\mathbf{C} \in \mathcal{AC}_X$ relative to X,*

$$X \perp_d \mathbf{V}^{\leq X} \backslash (De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})) \cup Pa(\mathbf{C})) \mid Pa(\mathbf{C}) \backslash \{X\}.$$

Proof. Since $\perp_d$ satisfies the composition and decomposition axioms, it suffices to show that $X \perp_d \{Y\} \mid Pa(\mathbf{C}) \setminus \{X\}$ for every $Y \in \mathbf{V}^{\leq X} \setminus (De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})) \cup Pa(\mathbf{C}))$.

Take a variable $Y \in \mathbf{V}^{\leq X} \setminus (De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})) \cup Pa(\mathbf{C}))$ and some path $\pi = (X, V_1, V_2, \ldots, V_n, Y)$ between X and Y in $\mathcal{G}$ for $n \geq 1$ (note that X, Y are non-adjacent by assumption). Let $\pi' = (X, V_1, \ldots, V_k)$ (with $k \leq n$) denote the longest sub-path of π starting from X, not including Y, that contains only bidirected edges. If $\pi' = \emptyset$, then $V_1 \prec X \implies V_1 \in Pa(\{X\}) \subseteq Pa(\mathbf{C})$, hence π is blocked. Otherwise, if for some $i \in [k]$, $V_i \notin An(\mathbf{C})$, then V_i blocks π. If every $V_i \in An(\mathbf{C})$, since $\mathbf{C}$ is an AC, the existence of π' implies $V_i \in \mathbf{C}$. Then, consider the subpath of π from V_k to Y. Note that V_k, Y are non-adjacent since $Y \notin De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})) \cup Pa(\mathbf{C})$. The sub-path has either $V_k \leftarrow V_{k+1} \circ\!\!-\!\!\circ$ or $V_k \rightarrow V_{k+1} \rightarrow$, both of which are blocked by $Pa(\mathbf{C}) \setminus \{X\}$. Therefore, π is blocked, and $X \perp_d \{Y\} \mid Pa(\mathbf{C}) \setminus \{X\}$. □

Theorem 1 (Equivalence of C-LMP and GMP). Let $\mathcal{G}$ be a causal graph and $\mathbf{V}^{\prec}$ a consistent ordering. A probability distribution over $\mathbf{V}$ satisfies the global Markov property for $\mathcal{G}$ if and only if it satisfies the c-component local Markov property for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$.

Proof. The proof is similar to that of (Lauritzen et al. 1990, Prop. 4) and (Richardson 2003, Thm. 2), which is based on the former.

($\implies$) Prop. C.1.1 shows that the CIs invoked by C-LMP are a subset of those invoked by GMP. Therefore, if a probability distribution $P(\mathbf{v})$ satisfies the GMP for a given DAG $\mathcal{G}$, it necessarily satisfies the C-LMP for $\mathcal{G}$ (with respect to any given ordering).

($\impliedby$) Next, we show that if a probability distribution $P(\mathbf{v})$ satisfies the C-LMP for a given DAG $\mathcal{G}$ with respect to a given ordering, it necessarily satisfies the GMP for $\mathcal{G}$. We show the other direction by induction on the number of nodes. Let I_k be the statement that for a graph $\mathcal{G}$ on k nodes, if a distribution $P(\mathbf{v})$ satisfies the C-LMP for $\mathcal{G}$, then it satisfies the GMP for $\mathcal{G}$. The base case is trivial. Assume for some k that I_j is true for all $j \leq k$. We will show this implies I_{k+1}. Fix a graph $\mathcal{G}$ with $k+1$ nodes, a consistent ordering $\prec$, and a distribution $P(\mathbf{v})$ which satisfies the C-LMP for $\mathcal{G}$ with respect to $\prec$. Consider a d-separation $\mathbf{X} \perp_d \mathbf{Y} \mid \mathbf{Z}$ in $\mathcal{G}$ for disjoint sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$. We need to show that $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$ in $P(\mathbf{v})$.

We claim we can assume, without loss of generality, that $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z} = \mathbf{V}$. First, we show how we can assume $An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}) = \mathbf{V}$. Consider $\mathcal{G}' = \mathcal{G}_{An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})}$, and let $\prec'$ be the ordering $\prec$ but removing variables in $\mathbf{V} \setminus An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$. Let $\mathbf{A} = An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})_{\mathcal{G}}$, so that $P(\mathbf{a}) = \sum_{\mathbf{v} \setminus \mathbf{a}} P(\mathbf{v})$. Since $\mathcal{G}'$ is a subgraph on an ancestral set, any AC in $\mathcal{G}'$ is an AC in $\mathcal{G}$, it is easy to see that if $P(\mathbf{v})$ satisfies the C-LMP for $\mathcal{G}$ with respect to $\prec$, then $P(\mathbf{a})$ satisfies the C-LMP for $\mathcal{G}$ with respect to $\prec'$. Since $\mathbf{X} \perp_d \mathbf{Y} \mid \mathbf{Z}$ in $\mathcal{G}$, and $\mathcal{G}'$ contains no more edges than $\mathcal{G}$, we also have $\mathbf{X} \perp_d \mathbf{Y} \mid \mathbf{Z}$ in $\mathcal{G}'$. By the inductive assumption for $\mathcal{G}'$, we have $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$ in $P(\mathbf{a})$, which implies $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$ in $P(\mathbf{v})$. Finally, we can extend $\mathbf{X}, \mathbf{Y}$ so that $An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}) = \mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$ (and reduce to the original separation statement using the decomposition axiom). For any $V \in An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}) \setminus \mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$, either $V \perp_d \mathbf{Y} \mid \mathbf{Z}$ or $V \perp_d \mathbf{X} \mid \mathbf{Z}$. Towards contradiction, assume V has an active path π_x to some node in $\mathbf{X}$ and an active path π_y to some node in $\mathbf{Y}$ when conditioning on $\mathbf{Z}$. Then, adjoining $\pi = \pi_x \cup \pi_y$ gives an active path between $\mathbf{X}$ and $\mathbf{Y}$ unless V is an inactive collider on this path. However, if $V \in An(\mathbf{X})$, the path $V \rightsquigarrow X$ to the descendant node $X \in \mathbf{X}$, adjoined with π_y, gives an active path between $\mathbf{X}$ and $\mathbf{Y}$ unless we condition on some descendant of V; the same applies if $V \in An(\mathbf{Y})$; and if $V \in An(\mathbf{Z})$, clearly, V is active when conditioning on $\mathbf{Z}$. We thus arrive at a contradiction.

Now, consider a separation $\mathbf{X} \perp_d \mathbf{Y} \mid \mathbf{Z}$ in $\mathcal{G}$ such that $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z} = \mathbf{V}$. We need to show that $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$ in $P(\mathbf{v})$. Let V^* be the final node in the ordering $\prec$ so that $\mathbf{V}^{\leq V^*} = \mathbf{V}$. Since $V \in \mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$, there are three cases to consider:

1. $V^* \in \mathbf{X}$.
   Since $\mathbf{X} \perp_d \mathbf{Y} \mid \mathbf{Z}$ in $\mathcal{G}$, we have $\mathbf{X} \setminus \{V^*\} \perp_d \mathbf{Y} \mid \mathbf{Z}$ in $\mathcal{G}_{\mathbf{V} \setminus \{V^*\}}$. Since $\mathcal{G}_{\mathbf{V} \setminus \{V^*\}}$ is ancestral, we apply a similar argument as in justifying the assumption that $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z} = \mathbf{V}$ to get, by the inductive assumption for $\mathcal{G}'$, that
   $$\mathbf{X} \setminus \{V^*\} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \text{ in } P(\mathbf{v}).$$
   Let $\mathbf{C} = \mathcal{C}(V^*)_{\mathcal{G}_{An(\mathbf{X} \cup \mathbf{Z})}}$. By C-LMP, we have $V^* \perp\!\!\!\perp \mathbf{V} \setminus (De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})) \cup Pa(\mathbf{C})) \mid Pa(\mathbf{C}) \setminus \{V^*\}$.
   First, note that $\mathbf{Y} \cap Pa(\mathbf{C}) = \emptyset$. Towards contradiction, assume that for some $Y \in \mathbf{Y}$, there is a path $\pi : Y \circ\!\!\rightarrow V_1 \leftrightarrow V_2 \leftrightarrow \ldots V_k \leftrightarrow V_{k+1} = V^*$ such that each $V_i \in \mathbf{C} \subseteq An(\mathbf{X} \cup \mathbf{Z})$. By induction on $i \in [k+1]$, we show that π is active when conditioning on $\mathbf{Z}$. For the base case, clearly, the subpath $Y \circ\!\!\rightarrow V_1$ of π is active when conditioned on $\mathbf{Z}$. Assume that, for some $i \in [k+1]$, the sub-path of π from Y to V_i is active. Consider the inductive step. If $V_i = V^*$, we are done. Otherwise, if $V_i \in An(\mathbf{Z})$, then V_i is active in π when conditioning on $\mathbf{Z}$. If $V_i \in An(\mathbf{X})$, then from the inductive assumption, there is a path from Y to V_i plus a path $V_i \rightsquigarrow X'$ to some $X' \in \mathbf{X}$ which is only blocked if $De(\{V_i\}) \cap \mathbf{Z} \neq \emptyset$. This again implies that V_1 is active in π when conditioning on $\mathbf{Z}$. In either case, the subpath of π from Y to V_{i+1} is active. This contradicts $\mathbf{X} \perp_d \mathbf{Y} \mid \mathbf{Z}$ in $\mathcal{G}$. Therefore, we can conclude $\mathbf{Y} \cap Pa(\mathbf{C}) = \emptyset$.
   Second, note that $\mathbf{Y} \cap De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})) = \emptyset$. This is because $Sp(\mathbf{C}) \setminus Pa(\mathbf{C}) = \emptyset$. For any $U \in Sp(\mathbf{C}), U \in \mathbf{X} \cup \mathbf{Z} \implies U \in \mathbf{C}$ by definition of $\mathbf{C}$. Therefore, $U \in Sp(\mathbf{C}) \setminus Pa(\mathbf{C}) \implies U \in \mathbf{Y} = \mathbf{V} \setminus \mathbf{X} \cup \mathbf{Z}$. However, for such U, there is a path $\pi : U \leftrightarrow V_1 \leftrightarrow V_2 \leftrightarrow \ldots V_k \leftrightarrow V_{k+1} = V^*$ with each $V_i \in An(\mathbf{X} \cup \mathbf{Z})$. By a similar induction as for the claim $\mathbf{Y} \cap Pa(\mathbf{C}) = \emptyset$, we can show that π is active when conditioning on $\mathbf{Z}$, which contradicts $\mathbf{X} \perp_d \mathbf{Y} \mid \mathbf{Z}$.
   We return to the CI statement
   $$V^* \perp\!\!\!\perp \mathbf{V} \setminus (De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})) \cup Pa(\mathbf{C})) \mid Pa(\mathbf{C}) \setminus \{V^*\}.$$
   Since $\mathbf{V} = \mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$ by assumption and $\mathbf{Y} \cap Pa(\mathbf{C}) = \mathbf{Y} \cap De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})) = \emptyset$, we can simplify this state-

ment to

$$V^* \perp\!\!\!\perp \mathbf{Y} \cup ((\mathbf{X} \cup \mathbf{Z}) \setminus (De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})) \cup Pa(\mathbf{C}))) \mid Pa(\mathbf{C}) \setminus \{V^*\}.$$

For any variable $W \in \mathbf{X} \cup \mathbf{Z}$, $W \in De(Sp(\mathbf{C}))$ implies there is some variable $B \in Sp(\mathbf{C}) \cap An(\{W\}) \subseteq Sp(\mathbf{C}) \cap An(\mathbf{X} \cup \mathbf{Z})$, hence $B \in \mathbf{C}$. Therefore, $De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})) = \emptyset$. This further implies

$$V^* \perp\!\!\!\perp \mathbf{Y} \cup ((\mathbf{X} \cup \mathbf{Z}) \setminus Pa(\mathbf{C})) \mid Pa(\mathbf{C}) \setminus \{V^*\}$$

By the weak union axiom, we get

$$V^* \perp\!\!\!\perp \mathbf{Y} \mid (\mathbf{X} \setminus \{V^*\}) \cup \mathbf{Z}$$

Applying the contraction axiom to $\mathbf{X} \setminus \{V^*\} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$ and $V^* \perp\!\!\!\perp \mathbf{Y} \mid (\mathbf{X} \setminus \{V^*\}) \cup \mathbf{Z}$ gives $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$ in $P(\mathbf{v})$.

2. $V^* \in \mathbf{Y}$. This is similar to the case $V \in \mathbf{X}$ (switching $\mathbf{X}, \mathbf{Y}$ in the proof).
3. $V^* \in \mathbf{Z}$. Since $\mathbf{X} \perp_d \mathbf{Y} \mid \mathbf{Z}$ in $\mathcal{G}$, we have $\mathbf{X} \perp_d \mathbf{Y} \mid \mathbf{Z} \setminus \{V^*\}$ in $\mathcal{G}_{\mathbf{V} \setminus \{V^*\}}$. Since $\mathcal{G}_{\mathbf{V} \setminus \{V^*\}}$ is a subgraph on an ancestral set, we apply a similar argument as in justifying the assumption that $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z} = \mathbf{V}$ to get, by the inductive assumption for $\mathcal{G}'$, that

$$\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \setminus \{V^*\} \text{ in } P(\mathbf{v}).$$

Let $\mathbf{C} = \mathcal{C}(V^*)_{\mathcal{G}}$. By C-LMP, we have $V^* \perp\!\!\!\perp \mathbf{V} \setminus (De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})) \cup Pa(\mathbf{C})) \mid Pa(\mathbf{C}) \setminus \{V^*\}$.
First, we show that either $Pa(\mathbf{C}) \cap \mathbf{Y} = \emptyset$ or $Pa(\mathbf{C}) \cap \mathbf{X} = \emptyset$. Assume, toward contradiction, that $Pa(\mathbf{C}) \cap \mathbf{Y} \neq \emptyset$ and $Pa(\mathbf{C}) \cap \mathbf{X} \neq \emptyset$. Then, there are variables $Y \in \mathbf{Y}, X \in \mathbf{X}$ and a path $\pi : Y \circ\!\!\rightarrow V_1 \leftrightarrow V_2 \leftrightarrow \ldots V_k \leftarrow\!\!\circ V_{k+1} = X$ for some $k \geq 0$. Let π' be a subpath of π such that one endpoint node of π' is in $\mathbf{X}$, the other endpoint node in $\mathbf{Y}$, and all intermediate nodes (if any) are in $\mathbf{Z}$. It is easy to see π' must exist since $\pi' = \pi$ if for each $i \in [k]$, we have $V_i \in \mathbf{Z}$; otherwise, we can construct π' by removing variables from π. Then, π' is active when conditioning on $\mathbf{Z}$, which contradicts $\mathbf{X} \perp_d \mathbf{Y} \mid \mathbf{Z}$.
Moreover, $Sp(\mathbf{C}) \setminus Pa(\mathbf{C}) = \emptyset$ because $\mathbf{C}$ is defined over $\mathbf{V} = \mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$. Thus, $\mathbf{X} \cap De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})) = \mathbf{Y} \cap De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})) = \emptyset$.
Return to the CI statement: $V^* \perp\!\!\!\perp \mathbf{V} \setminus (De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})) \cup Pa(\mathbf{C})) \mid Pa(\mathbf{C}) \setminus \{V^*\}$. If $Pa(\mathbf{C}) \cap \mathbf{Y} = \emptyset$, this simplifies to $V^* \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{X} \cup (\mathbf{Z} \setminus \{V^*\})$ by an argument similar to Case (1). The contraction axiom applied to $V^* \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{X} \cup (\mathbf{Z} \setminus \{V^*\})$ and $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \setminus \{V^*\}$ gives $\mathbf{X} \cup \{V^*\} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \setminus \{V^*\}$. Applying the weak union axiom to this last CI, we get $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$. A similar argument applies if $Pa(\mathbf{C}) \cap \mathbf{X} = \emptyset$.

□

Corollary 1 (Equivalence of C-LMP and the Ordered Local Markov Property). Let $\mathcal{G}$ be a causal graph and $\mathbf{V}^{\prec}$ a consistent ordering. A probability distribution over $\mathbf{V}$ satisfies the ordered local Markov property for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$ if and only if it satisfies the c-component local Markov property for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$.

Proof. By Thm. 1, a probability distribution $P(\mathbf{v})$ over $\mathbf{V}$ satisfies the C-LMP for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$ if and only if it satisfies the GMP for $\mathcal{G}$. By (Richardson 2003, Thm. 2, Section 3.1), a probability distribution $P(\mathbf{v})$ over $\mathbf{V}$ satisfies the ordered local Markov property for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$ if and only if it satisfies the GMP for $\mathcal{G}$. □

Theorem 2 (Unique AC for each CI Invoked by C-LMP). Let $\mathcal{G}$ be a causal graph, $\mathbf{V}^{\prec}$ a consistent ordering, and X a variable in $\mathbf{V}^{\prec}$. For every conditional independence relation invoked by the c-component local Markov property of the form $X \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z}$, there is exactly one ancestral c-component $\mathbf{C} \in \mathcal{AC}_X$ such that $\mathbf{W} = \mathbf{V}^{\leq X} \setminus ((De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C}))) \cup Pa(\mathbf{C}))$ and $\mathbf{Z} = Pa(\mathbf{C}) \setminus \{X\}$.

Proof. The result follows from Def. 5, Lemma B.3.1, and Cor. B.3.1. □

Proposition 1 (Number of CIs Invoked by C-LMP). Given a causal graph $\mathcal{G}$ and a consistent ordering $\mathbf{V}^{\prec}$, let n and $s \leq n$ denote the number of variables and the size of the largest c-component in $\mathcal{G}$ respectively. Then, the c-component local Markov property for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$ invokes $O(n2^s)$ conditional independencies implied by $\mathcal{G}$ over $\mathbf{V}$. Moreover, there exists a graph $\mathcal{G}$ and a consistent ordering $\mathbf{V}^{\prec}$ for which the property induces $\Omega(2^n)$ conditional independencies.

Proof. By Def. 5, the set of CIs invoked by C-LMP for a variable $X \in \mathbf{V}^{\prec}$ is in bijection with the set of ACs, $\mathcal{AC}_X$. Therefore, it suffices to bound $|\mathcal{AC}_X|$. Recall that $\mathcal{AC}_X \subseteq \mathcal{P}(\mathcal{C}(X)_{\mathcal{G}})$ (where $\mathcal{P}(\cdot)$ denotes the power-set operation). Then, $|\mathcal{C}(X)_{\mathcal{G}}| \leq s \implies |\mathcal{P}(\mathcal{C}(X)_{\mathcal{G}})| \leq 2^s \implies |\mathcal{AC}_X| \leq 2^s$. Total number of CIs k invoked by C-LMP for all variables is thus $k \leq n2^s \in O(n2^s)$.

Next, consider the graph $\mathcal{G}$ shown in Fig. B.4.2 for which C-LMP invokes $\Omega(2^n)$ CIs.

Fix $V_i, i \in [k]$. For each $\mathbf{C} \subseteq \{V_j\}_{j<i}$, we get an AAC $\{V_i, X, Z\} \cup \mathbf{C}$ relative to V_i inducing the CI: $V_i \perp\!\!\!\perp \{Y\} \mid \{X, Z\} \cup \mathbf{C}$ (The definition of admissibility of AC is given by Def. 6). There are 2^{i-1} such CIs for each V_i. Then, the total number of CIs across all such V_i is

$$\sum_{i=1}^{k} 2^{i-1} = 2^{k-1} - 1 = 2^{n-4} - 1 \in \Omega(2^n)$$

Since, for any $\mathcal{G}$, we have that $s \leq n$, the upper bound $O(n2^s)$ is thus tight ignoring the linear term in n. □

C.2 Section 4 Proofs

Notation. For the proofs in this section, given a causal graph $\mathcal{G}$ defined on a set of variables $\mathbf{V}$, and variables $X, Y \in \mathbf{V}$, we use $X \sim Y$ to denote an arbitrary path (possibly of length 0, when $X = Y$) between X and Y in $\mathcal{G}$; $X \rightsquigarrow Y$ to denote a directed path (possibly of length 0, when $X = Y$) from X to Y in $\mathcal{G}$; and $X \circ\!\!\rightarrow Y$ to denote that there is either an edge $X \rightarrow Y$ or $X \leftrightarrow Y$ in $\mathcal{G}$.

Proposition C.2.1 (Time Complexity of Computing a C-component)**.** *Given a causal graph $\mathcal{G}$ over a set of variables $\mathbf{V}$ and a variable $X \in \mathbf{V}$, the c-component $\mathcal{C}(X)_{\mathcal{G}}$ containing X in $\mathcal{G}$ is computable in time $O(n + m)$, where n and m are the numbers of nodes and edges in $\mathcal{G}$ respectively.*

1: **function** ISADMISSIBLE($\mathcal{G}_{\mathbf{V}^{\leq X}}, X, \mathbf{V}^{\leq X}, \mathbf{C}$)
2: **Output:** True if a given AC $\mathbf{C}$ relative to X is admissible; False otherwise.
3: $\mathbf{S}^+ \leftarrow \mathbf{V}^{\leq X} \setminus De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C}))$
4: $\mathbf{W} \leftarrow \mathbf{S}^+ \setminus Pa(\mathbf{C})$
5: **if** $\mathbf{W} \neq \emptyset$ **then**
6: **return** True
7: **else**
8: **return** False
9: **end function**

Figure C.2.1: A function that checks if a given AC is admissible.

Proof. Using breadth-first search (BFS), compute the set of nodes reachable from the starting node X by following only bidirected edges. This takes time $O(n+m)$, the complexity of BFS. □

Lemma C.2.1 (Correctness of ISADMISSIBLE)**.** *Given a causal graph $\mathcal{G}_{\mathbf{V}^{\leq X}}$, a variable X, and a set of variables $\mathbf{V}^{\leq X}$, let $\mathbf{C}$ be an ancestral c-component relative to X. Then,* ISADMISSIBLE *returns True if $\mathbf{C}$ is admissible, and False otherwise.* ISADMISSIBLE *takes $O(n+m)$ time where n and m represent the number of nodes and edges in $\mathcal{G}$, respectively.*

Proof. For correctness, it immediately follows from Def. 5 and Def. 6.

ISADMISSIBLE runs in $O(n+m)$ time since the construction of the sets $\mathbf{S}^+$ and $\mathbf{W}$ takes $O(n+m)$ time for each set. □

Lemma C.2.2 (Existence of a Separator)**.** *Given a causal graph $\mathcal{G}$, let $\mathbf{I}, \mathbf{R}, \mathbf{X}, \mathbf{Y}$ be sets of nodes with $\mathbf{I} \subseteq \mathbf{R}$ and $\mathbf{R} \cap (\mathbf{X} \cup \mathbf{Y}) = \emptyset$. If there exists a set $\mathbf{Z}_0$ separating $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}$ such that $\mathbf{I} \subseteq \mathbf{Z_0} \subseteq \mathbf{R}$, then $\mathbf{Z} = An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})_{\mathcal{G}} \cap \mathbf{R}$ is such a set.*

Proof. Assume there exists $\mathbf{Z}_0$ separating $\mathbf{X}, \mathbf{Y}$ such that $\mathbf{I} \subseteq \mathbf{Z_0} \subseteq \mathbf{R}$. For some $X \in \mathbf{X}$, $Y \in \mathbf{Y}$, consider a path π from X to Y in $\mathcal{G}$, consisting of nodes $\{X = V_0, V_1, \ldots, V_n, V_{n+1} = Y\}$ where $V_i, V_i + 1$ are adjacent in $\mathcal{G}$ for $0 \leq i \leq n$. Note that we must have $n \geq 1$; otherwise, X, Y are adjacent and cannot be separated.

If none of the variables $V_i, i \in [n]$ is a collider, then each V_i must be in $An(\{X, Y\})$. If $V_i \notin \mathbf{R}$ for any $i \in [n]$, then $V_i \notin \mathbf{Z_0} \subseteq \mathbf{R}$ for any $i \in [n]$, and hence $\mathbf{Z}_0$ does not block π, which is a contradiction. Therefore, there exists V_i such that $V_i \in An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})_{\mathcal{G}} \cap \mathbf{R} = \mathbf{Z}$ and hence $\mathbf{Z}$ blocks π.

If some V_i is a collider, let $\mathbf{C} = \{C_{i_1}, \ldots, C_{i_k}\} \subseteq \{V_1, \ldots, V_n\}$ denote the set of colliders on π such that $i_j < i_{j+1}$ for $1 \leq j \leq k-1$. If there is a variable $C \in \mathbf{C}$ such that $\mathbf{Z} \cap De(\{C\})_{\mathcal{G}} = \emptyset$ (in other words, $\mathbf{Z}$ does not contain C or any of its descendants), then $\mathbf{Z}$ blocks π due to the inactive collider C. Otherwise, consider the case that $\mathbf{C} \subseteq An(\mathbf{Z})_{\mathcal{G}}$ i.e. for every $C \in \mathbf{C}$, either C is in $\mathbf{Z}$ or some descendant of C is in $\mathbf{Z}$, and hence C is active (when conditioning on $\mathbf{Z}$. Since $\mathbf{Z} = An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})_{\mathcal{G}} \cap \mathbf{R} \subseteq An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})_{\mathcal{G}}$, we have $\mathbf{C} \subseteq An(\mathbf{Z})_{\mathcal{G}} \implies An(\mathbf{C})_{\mathcal{G}} \subseteq An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})_{\mathcal{G}}$. For any V_i in π, either $V_i \in \mathbf{C}$ or $V_i \in An(\{X, Y\})$ or $V_i \in An(\mathbf{C})$; therefore, $\{V_i\}_{1:n} \subseteq An(\{X, Y\} \cup \mathbf{C})_{\mathcal{G}} \subseteq An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})_{\mathcal{G}}$. Hence, π is blocked by $\mathbf{Z}$ unless every $V_i \in \mathbf{R}$ is a collider; that is, $\{V_i\}_{1:n} \cap \mathbf{R} \subseteq \mathbf{C}$. Assume toward contradiction that $\{V_i\}_{1:n} \cap \mathbf{R} \subseteq \mathbf{C}$.

We show, by induction on the index $i_j, j \in [k]$ of $\mathbf{C}$, that for every $j \in [k]$, there exists a variable $X_0 \in \mathbf{X}$ such that there is an active $X_0 \sim C_{i_k} \leftarrow \circ V_{i_j+1}$ path when conditioning on $\mathbf{Z_0}$.

Base case. Consider $C_{i_1} \in An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})_{\mathcal{G}}$. The sub-path of π from X to C_{i_1} is unblocked by $\mathbf{Z_0} \subseteq \mathbf{R}$. This is because for any node V on this sub-path (excluding X and C_{i_1}), $V \notin \mathbf{C}$ by assumption and $\{V_i\}_{1:n} \cap \mathbf{R} \subseteq \mathbf{C} \implies V \notin \mathbf{R} \implies V \notin \mathbf{Z}_0$.

- If $C_{i_1} \in An(\mathbf{Y})_{\mathcal{G}}$, there is a directed path π' from C_{i_1} to Y' for some $Y' \in \mathbf{Y}$. The $X \sim C_{i_1}$ sub-path of π (which is unblocked by $\mathbf{Z}_0$), adjoined with π', gives an active path from $X \in \mathbf{X}$ to $Y' \in \mathbf{Y}$. For $\mathbf{Z}_0$ to block this path, it must block π'. Hence, $\mathbf{Z_0}$ contains a descendant of C_{i_1} and C_{i_1} is active when conditioning on $\mathbf{Z_0}$, giving an active sub-path of π, $X \sim V_{i_1-1} \circ \rightarrow C_{i_1} \leftarrow \circ V_{i_1+1}$.
- If $C_{i_1} \in An(\mathbf{X})_{\mathcal{G}}$, there is a directed path π' from C_{i_1} to X' for some $X' \in \mathbf{X}$. If $\mathbf{Z_0}$ contains a descendant of C_{i_1}, then C_{i_1} is active when conditioning on $\mathbf{Z_0}$. Therefore, the sub-path of π, $X \sim V_{i_1-1} \circ \rightarrow C_{i_1} \leftarrow \circ V_{i_1+1}$ is active. If $\mathbf{Z_0}$ contains no descendants of C_{i_1}, then π' is unblocked by $\mathbf{Z_0}$, giving an active $X' \leftsquigarrow C_{i_1} \leftarrow \circ V_{i_1+1}$ path.
- If $C_{i_1} \in An(\mathbf{I})_{\mathcal{G}}$, since $\mathbf{I} \subseteq \mathbf{Z_0}$, we condition on a descendant of C_{i_1} and C_{i_1} is active, giving an active sub-path of π, $X \sim V_{i_1-1} \circ \rightarrow \mathcal{C}_{i_1} \leftarrow \circ V_{i_1+1}$.

Inductive assumption. Assume for some $j \in [k-1]$, there is an active $X_0 \sim \mathbf{C}_{i_j} \leftarrow \circ V_{i_j+1}$ path for some $X_0 \in \mathbf{X}$.

Inductive step. We show this implies the existence of an active $X_0' \sim \mathbf{C}_{i_{j+1}} \leftarrow \circ V_{i_{j+1}+1}$ path for some $X_0' \in \mathbf{X}$. Note that the sub-path of π from C_{i_j} to $C_{i_{j+1}}$ is unblocked by $\mathbf{Z_0}$. This is because for any node V on this sub-path (excluding C_j and C_{j+1}), $V \notin \mathbf{C}$ by assumption and $\{V_i\}_{1:n} \cap \mathbf{R} \subseteq \mathbf{C} \implies V \notin \mathbf{R} \implies V \notin \mathbf{Z}_0$.

- If $C_{i_{j+1}} \in An(\mathbf{Y})_{\mathcal{G}}$, there is a directed path $\pi' : C_{i_{j+1}} \rightsquigarrow Y'$ for some $Y' \in \mathbf{Y}$. By the inductive assumption, we get an active path $X_0 \sim \mathbf{C}_{i_j} \leftarrow \circ V_{i_j+1} \circ \rightarrow C_{i_{j+1}} \rightsquigarrow Y'$. For $\mathbf{Z_0}$ to block this path, it must block π'. Hence, $\mathbf{Z_0}$ contains a descendant of $C_{i_{j+1}}$ and $C_{i_{j+1}}$ is active when conditioning on $\mathbf{Z_0}$, giving an active $X_0 \sim C_{i_j} \leftarrow \circ V_{i_j+1} \sim V_{i_{j+1}-1} \circ \rightarrow C_{i_{j+1}} \leftarrow \circ V_{i_{j+1}+1}$ path.
- If $C_{i_{j+1}} \in An(\mathbf{X})_{\mathcal{G}}$, there is a directed path π' from $C_{i_{j+1}}$ to X' for some $X' \in \mathbf{X}$. If $\mathbf{Z_0}$ contains a descendant of $C_{i_{j+1}}$, then $C_{i_{j+1}}$ is active when conditioning on $\mathbf{Z_0}$. Using the inductive assumption, we get an active path $X_0 \sim C_{i_j} \leftarrow \circ V_{i_j+1} \sim V_{i_{j+1}-1} \circ \rightarrow C_{i_{j+1}} \leftarrow \circ V_{i_{j+1}+1}$. If $\mathbf{Z_0}$ contains no descendants of $C_{i_{j+1}}$, then π' is unblocked by $\mathbf{Z_0}$, giving an active path $X' \leftsquigarrow C_{i_{j+1}} \leftarrow \circ V_{i_{j+1}+1}$ path.
- If $C_{i_{j+1}} \in An(\mathbf{I})_{\mathcal{G}}$, since $\mathbf{I} \subseteq \mathbf{Z_0}$, we condition on a descendant of $C_{i_{j+1}}$ and $C_{i_{j+1}}$ is active. Using the inductive assumption, we get an active path $X_0 \sim C_{i_j} \leftarrow \circ V_{i_j+1} \sim V_{i_{j+1}-1} \circ \rightarrow C_{i_{j+1}} \leftarrow \circ V_{i_{j+1}+1}$.

1: **function** FINDSEPARATOR($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$)
2: **Output:** A set of variables $\mathbf{Z}$ d-separating $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}$ under the constraint $\mathbf{I} \setminus (\mathbf{X} \cup \mathbf{Y}) \subseteq \mathbf{Z} \subseteq \mathbf{R} \setminus (\mathbf{X} \cup \mathbf{Y})$ if such $\mathbf{Z}$ exists; $\perp$ otherwise.
3: $\mathbf{R}' \leftarrow \mathbf{R} \setminus (\mathbf{X} \cup \mathbf{Y})$
4: $\mathbf{Z} \leftarrow An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})_{\mathcal{G}} \cap \mathbf{R}'$
5: **if** $\mathbf{Z}$ d-separates $\mathbf{X}, \mathbf{Y}$ in $\mathcal{G}$ **then**
6: **return** $\mathbf{Z}$
7: **else**
8: **return** $\perp$
9: **end function**

Figure C.2.2: A function that finds a separator of a given pair of sets of variables, if it exists.

By induction, we have an active $X_0 \sim C_{i_k} \leftarrow \circ V_{i_k+1}$ path for some $X_0 \in \mathbf{X}$. The $V_{i_k+1} \sim Y$ sub-path of π is active when conditioning on $\mathbf{Z_0}$. This is because for any node V on this sub-path (excluding V_{i_k+1} and Y), $V \notin \mathbf{C}$ by assumption and $\{V_i\}_{1:n} \cap \mathbf{R} \subseteq \mathbf{C} \implies V \notin \mathbf{R} \implies V \notin \mathbf{Z}_0$. Recall that by assumption, V_{i_k+1} is not a collider. We thus have an $X_0 \sim Y$ path which is active when conditioning on $\mathbf{Z_0}$. We thus have a contradiction. □

Lemma C.2.3 (Correctness of FINDSEPARATOR)**.** *Given a causal graph $\mathcal{G}$, let $\mathbf{I}, \mathbf{R}, \mathbf{X}, \mathbf{Y}$ be sets of nodes with $\mathbf{I} \subseteq \mathbf{R}$.* FINDSEPARATOR*$(\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R})$ has a non-empty output if and only if there exists a set $\mathbf{Z}$ separating $\mathbf{X}, \mathbf{Y}$ in $\mathcal{G}$ such that $\mathbf{I} \setminus (\mathbf{X} \cup \mathbf{Y}) \subseteq \mathbf{Z} \subseteq \mathbf{R} \setminus (\mathbf{X} \cup \mathbf{Y})$. Moreover, any output $\mathbf{Z} \neq \perp$ satisfies $\mathbf{X} \perp_{\mathcal{G}} \mathbf{Y} \mid \mathbf{Z}$ and $\mathbf{I} \setminus (\mathbf{X} \cup \mathbf{Y}) \subseteq \mathbf{Z} \subseteq \mathbf{R} \setminus (\mathbf{X} \cup \mathbf{Y})$. Finally,* FINDSEPARATOR *runs in time $O(n + m)$, where n and m are the numbers of nodes and edges respectively in $\mathcal{G}$.*

Proof. The correctness is immediate from the construction of FINDSEPARATOR and Lemma C.2.2. For the runtime, constructing $\mathbf{R}'$ and $\mathbf{Z}$ in the algorithm takes time $O(n)$ and $O(n + m)$ respectively. Verifying whether $\mathbf{Z}$ d-separates $\mathbf{X}$ from $\mathbf{Y}$ in $\mathcal{G}$ or not, as shown in line 5, may be performed by using the Bayes-Ball algorithm (Shachter 2013) on a modified graph $\mathcal{G}'$ of $\mathcal{G}$ where $\mathcal{G}'$ is constructed as follows: start from $\mathcal{G}' = \mathcal{G}$, and replace each edge $X \leftrightarrow Y$ with an explicit latent common cause $X \leftarrow U_{XY} \rightarrow Y$. The construction of $\mathcal{G}'$ takes $O(n + m)$ time, and the Bayes-Ball algorithm runs in $O(n + m)$ time. The overall runtime of FINDSEPARATOR is thus $O(n + m)$. □

Since the size of the input graph $\mathcal{G}$ is $O(n + m)$, FINDSEPARATOR is asymptotically optimal.

Lemma 1 (Correctness of FINDAAC). Given a causal graph $\mathcal{G}$, a consistent ordering $\mathbf{V}^{\prec}$, and a variable $X \in \mathbf{V}^{\prec}$, let $\mathbf{I}, \mathbf{R}$ be ancestral c-components relative to X such that $\mathbf{I} \subseteq \mathbf{R}$. FINDAAC$(\mathcal{G}_{\mathbf{V}^{\leq X}}, X, \mathbf{V}^{\leq X}, \mathbf{I}, \mathbf{R})$ outputs an admissible ancestral c-component $\mathbf{C}$ relative to X such that $\mathbf{I} \subseteq \mathbf{C} \subseteq \mathbf{R}$ if such a $\mathbf{C}$ exists, and $\perp$ otherwise.

Proof. By assumption, $\mathbf{I}$ is an AC relative to X in the desired range since $\mathbf{I} \subseteq \mathbf{I} \subseteq \mathbf{R}$. FINDAAC outputs $\mathbf{I}$ (at line 4) if and only if $\mathbf{I}$ is admissible. This follows from the correctness of ISADMISSIBLE (by Lemma C.2.1).

Assume $\mathbf{I}$ is not admissible. It remains to show that there exists an AAC $\mathbf{C_0}$ relative to X such that $\mathbf{I} \subsetneq \mathbf{C_0} \subseteq \mathbf{R}$ if and only if there exists a variable $D \in De(Sp(\mathbf{I}) \setminus Pa(\mathbf{I}))$ and a set $\mathbf{Z}$ such that $Pa(\mathbf{I}) \setminus \{X, D\} \subseteq \mathbf{Z} \subseteq Pa(\mathbf{R}) \setminus \{X, D\}$ and $X \perp_{\mathcal{G}} D \mid \mathbf{Z}$. Moreover, the output of FINDAAC at line 8 must be an AAC relative to X in the given range.

($\implies$) Since $\mathbf{I}$ is not admissible, by Def. 6 we have

$$\mathbf{V}^{\leq X} \setminus (Pa(\mathbf{I}) \cup De(Sp(\mathbf{I}) \setminus Pa(\mathbf{I}))) = \emptyset \tag{16}$$

Since $\mathbf{C_0} \supsetneq \mathbf{I}$ is admissible, by Def. 6 we have

$$\mathbf{V}^{\leq X} \setminus (Pa(\mathbf{C_0}) \cup De(Sp(\mathbf{C_0}) \setminus Pa(\mathbf{C_0}))) \neq \emptyset \tag{17}$$

However, $\mathbf{I} \subsetneq \mathbf{C_0} \implies Pa(\mathbf{I}) \subseteq Pa(\mathbf{C_0})$. Therefore, Eq. (16) and Eq. (17) imply that there exists a variable $D \in \mathbf{V}^{\leq X}$ such that $D \in De(Sp(\mathbf{I}) \setminus Pa(\mathbf{I}))$ and $d \notin Pa(\mathbf{C_0}) \cup De(Sp(\mathbf{C_0}) \setminus Pa(\mathbf{C_0}))$. By the definition of C-LMP (shown in Def. 5), we have that $X \perp_{\mathcal{G}} D \mid Pa(\mathbf{C_0}) \setminus \{X\}$. Since $\mathbf{I} \subseteq \mathbf{C_0} \subseteq \mathbf{R}$ and $D \notin Pa(\mathbf{C_0})$, we have $Pa(\mathbf{I}) \setminus \{X, D\} \subseteq Pa(\mathbf{C_0}) \setminus \{X\} \subseteq Pa(\mathbf{R}) \setminus \{X, D\}$. Therefore, $\mathbf{Z} = Pa(\mathbf{C_0}) \setminus \{X\}$ is a set such that $Pa(\mathbf{I}) \setminus \{X, D\} \subseteq \mathbf{Z} \subseteq Pa(\mathbf{R}) \setminus \{X, D\}$ and $X \perp_{\mathcal{G}} D \mid \mathbf{Z}$. The correctness of FINDSEPARATOR (Lemma C.2.3) implies that FINDAAC detects the existence of $\mathbf{Z}$ and outputs $\mathbf{C} = \mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{I} \cup \mathbf{Z})}}$ at line 8. In the proof for the reverse direction, we will show that $\mathbf{C}$ thus defined is in fact admissible.

($\impliedby$) Consider some $D \in De(Sp(\mathbf{I}) \setminus Pa(\mathbf{I}))$ such that

$$\mathbf{Z} = \text{FINDSEPARATOR}(\mathcal{G}_{\mathbf{V}^{\leq X}}, \{X\}, \{D\}, Pa(\mathbf{I}), Pa(\mathbf{R})) \neq \perp$$

By the correctness of FINDSEPARATOR (Lemma C.2.3), we have $X \perp_{\mathcal{G}} D \mid \mathbf{Z}$. We give a constructive proof of existence by showing that $\mathbf{C} = \mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{I} \cup \mathbf{Z})}}$ is an AAC relative to X such that $\mathbf{I} \subsetneq \mathbf{C} \subseteq \mathbf{R}$.

Clearly, $\mathbf{C}$ is an AC by construction and $\mathbf{I} = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{I}}} \subseteq \mathbf{C}$. Moreover,

$$\begin{aligned}
\mathbf{I}, \mathbf{Z} \subseteq Pa(\mathbf{R}) &\implies \mathbf{I} \cup \mathbf{Z} \subseteq Pa(\mathbf{R}) && (18)\\
&\implies An(\mathbf{I} \cup \mathbf{Z}) \subseteq An(\mathbf{R}) && (19)\\
&\implies \mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{I} \cup \mathbf{Z})}} \subseteq \mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{R})}} && (20)\\
&\implies \mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{I} \cup \mathbf{Z})}} \subseteq \mathbf{R} && (21)
\end{aligned}$$

where the last implication follows since $\mathbf{R}$ is an AC relative to X by assumption, implying that $\mathbf{R} = \mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{R})}}$. Moreover, we claim that $\mathbf{C}$ is admissible, i.e.,

$$\mathbf{S}^{+} = \mathbf{V}^{\leq X} \setminus (Pa(\mathbf{C}) \cup De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C}))) \neq \emptyset$$

We will show that $D \notin Pa(\mathbf{C}) \cup De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C}))$, hence $\mathbf{S}^{+}$ contains D (and is therefore non-empty). We know $D \in \mathbf{V}^{\leq X}$. Assume, towards contradiction, that $D \in Pa(\mathbf{C}) \cup De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C}))$. Note that $\mathbf{I} \setminus \{X, D\} \subseteq \mathbf{Z} \implies An(\mathbf{I}) \subseteq An(\mathbf{Z} \cup \{X, D\})$. Therefore, $\mathbf{C} \subseteq An(\mathbf{I} \cup \mathbf{Z}) \subseteq An(\mathbf{Z} \cup \{X, D\})$. Since X, D are non-adjacent (because $\mathbf{Z}$ separates them), this implies the existence of a path π of one of the following types:

1. If $D \in \mathbf{C}$, then $X \leftrightarrow V_1 \cdots \leftrightarrow V_n \leftrightarrow V_{n+1} = D$ with $n \geq 1$ and each $V_i \in An(\mathbf{Z} \cup \{X, D\})$ for $i \in [n + 1]$

2. If $D \in Pa(\mathbf{C})$, then $X \leftrightarrow V_1 \cdots \leftrightarrow V_n \leftarrow V_{n+1} = D$ with $n \geq 1$ and each $V_i \in An(\mathbf{I} \cup \mathbf{Z})$ for $i \in [n+1]$
3. If $D \in De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})))$, then $X \leftrightarrow V_1 \leftrightarrow \cdots \leftrightarrow V_n \leftrightarrow V_{n+1} = A \rightsquigarrow D$ with each $V_i \in An(\mathbf{Z} \cup \{X, D\})$ for each $i \in [n+1]$ with $n \geq 0$ and $A \in Sp(\mathbf{C}) \setminus Pa(\mathbf{C})$. It is possible that the path $A \rightsquigarrow d$ has length 0, i.e., $A = D$.

We show by induction that there is an active $X \sim V_i \leftarrow \circ V_{i+1}$ path for each $i \in [n]$ when conditioning on $\mathbf{Z}$.

Base case. We know $V_1 \in An(\mathbf{Z} \cup \{X, D\})$.

- If $V_1 \in An(\mathbf{Z})$, V_1 is active when conditioning on $\mathbf{Z}$, hence the sub-path $X \leftrightarrow V_1 \leftarrow \circ V_2$ of π is active.
- If $V_1 \in An(\{D\})$, then there is a path $X \leftrightarrow V_1 \rightsquigarrow D$. Since $\mathbf{Z}$ must block this path, we have $\mathbf{Z} \cap De(\{V_1\}) \neq \emptyset$, hence V_1 is active when conditioning on $\mathbf{Z}$ and the sub-path $X \leftrightarrow V_1 \leftarrow \circ V_2$ of π is active.
- If $V_1 \in An(\{X\})$, then $X \in \mathbf{I}$, $V_1 \in Sp(\{X\})$, and $\mathbf{I}$ is an AC implies that $V_1 \in \mathbf{I}$. Since $V_1 \notin \{X, D\}$, this implies that $V_1 \in \mathbf{Z}$. Therefore, V_1 is active when conditioning on $\mathbf{Z}$ and the sub-path $X \leftrightarrow V_1 \leftarrow \circ V_2$ of π is active.

Inductive assumption. Assume, for some $i \in [n]$, there is an active $X \sim V_i \leftrightarrow V_{i+1}$ path when conditioning on $\mathbf{Z}$.

Inductive step. We show that there is an active $X \sim V_{i+1} \leftarrow \circ V_{i+2}$ path when conditioning on $\mathbf{Z}$. We know $V_{i+1} \in An(\mathbf{Z} \cup \{X, D\})$.

- If $V_{i+1} \in An(\mathbf{Z})$, V_{i+1} is active when conditioning on $\mathbf{Z}$. Therefore, the inductive assumption gives us an active path $X \sim V_i \leftrightarrow V_{i+1} \leftarrow \circ V_{i+2}$ when conditioning on $\mathbf{Z}$.
- If $V_{i+1} \in An(\{D\})$, then there is a path $V_{i+1} \rightsquigarrow D$. By the inductive assumption, there is an active path $X \sim V_i \leftrightarrow V_{i+1}$ when conditioning on $\mathbf{Z}$. Since $\mathbf{Z}$ must block the path $X \sim V_i \leftrightarrow V_{i+1} \rightsquigarrow D$, we have $\mathbf{Z} \cap De(\{V_{i+1}\}) \neq \emptyset$, hence V_{i+1} is active when conditioning on $\mathbf{Z}$ and the path $X \sim V_i \leftrightarrow V_{i+1} \leftarrow \circ V_{i+2}$ is active.
- If $V_{i+1} \in An(\{X\})$, then there is a path $X \leftsquigarrow V_{i+1}$. If $De(\{V_{i+1}\}) \cap \mathbf{Z} \neq \emptyset$, then V_{i+1} is active when conditioning on $\mathbf{Z}$ and by the inductive assumption, the path $X \sim V_i \leftrightarrow V_{i+1} \leftarrow \circ V_{i+2}$ is active. If $De(\{V_{i+1}\}) \cap \mathbf{Z} = \emptyset$, then the path $X \leftsquigarrow V_{i+1} \leftarrow \circ V_{i+2}$ is active.

Therefore, by induction, there is an active $X \sim V_n \leftarrow \circ V_{n+1}$ path when conditioning on $\mathbf{Z}$. If $V_{n+1} = D$, this contradicts $X \perp_{\mathcal{G}} d \mid \mathbf{Z}$. Otherwise, if $V_{n+1} = A \rightsquigarrow D$ in Case (3), then $\mathbf{Z}$ must block the path $A \rightsquigarrow D$. This implies that $A \in An(\mathbf{Z})$; moreover, $A \in Sp(\mathbf{C})$ and $\mathbf{C} = \mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{I} \cup \mathbf{Z})}}$ implies that $A \in \mathbf{C}$, which contradicts the assumption that $A \in Sp(\mathbf{C}) \setminus Pa(\mathbf{C})$. □

Proposition C.2.2 (Runtime of FINDAAC)**.** *Given a causal graph $\mathcal{G}$, a consistent ordering $\mathbf{V}^{\prec}$, and a variable $X \in \mathbf{V}^{\prec}$, let $\mathbf{I}, \mathbf{R}$ be ancestral c-components relative to X such that $\mathbf{I} \subseteq \mathbf{R}$.* FINDAAC*$(\mathcal{G}_{\mathbf{V} \leq X}, X, \mathbf{V}^{\leq X}, \mathbf{I}, \mathbf{R})$ runs in $O(n(n+m))$ time where n and m denote the numbers of nodes and edges in $\mathcal{G}$ respectively.*

Proof. A call to the function ISADMISSIBLE in line 3 takes $O(n+m)$ time (by Lemma C.2.1). FINDAAC computes a set of variables $De(Sp(\mathbf{I}) \setminus Pa(\mathbf{I}))$ (shown in line 5) only once, which takes $O(n+m)$ time. There are at most $O(n)$ iterations of the for loop, within which a call to the function FINDSEPARATOR (by Lemma C.2.3) and the construction of a c-component $\mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{I} \cup \mathbf{Z})}}$ in line 8 (by Prop. C.2.1) take time $O(n+m)$. Thus, the total runtime of FINDAAC is $O(n(n+m))$. □

Proposition C.2.3 (Ancestrality of Modified ACs)**.** *Given a causal graph $\mathcal{G}$, a consistent ordering $\mathbf{V}^{\prec}$, and a variable $X \in \mathbf{V}^{\prec}$, let $\mathbf{C}$ be an ancestral c-component relative to X. For any $\mathbf{S} \subseteq \mathbf{V}^{\prec}$ such that $X \notin De(\mathbf{S})$, $\mathbf{C}_{\mathbf{S}} = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{C} \setminus De(\mathbf{S})}}$ is an ancestral c-component relative to X.*

Proof. It suffices to show that $\mathbf{C}_{\mathbf{S}} = \mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{C}_{\mathbf{S}})}}$. Since $\mathbf{C}_{\mathbf{S}} \subseteq An(\mathbf{C}_{\mathbf{S}})$ and $\mathbf{C}_{\mathbf{S}} = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{C}_{\mathbf{S}}}}$, we have $\mathbf{C}_{\mathbf{S}} \subseteq \mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{C}_{\mathbf{S}})}}$. To show $\mathbf{C}_{\mathbf{S}} \supseteq \mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{C}_{\mathbf{S}})}}$, we make use of two facts. Since $\mathbf{C}_{\mathbf{S}} = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{C} \setminus De(\mathbf{S})}}$, we have $\mathbf{C}_{\mathbf{S}} \cap De(\mathbf{S}) = \emptyset$. This further implies that $An(\mathbf{C}_{\mathbf{S}}) \cap De(\mathbf{S}) = \emptyset$ (if some $W \in An(\mathbf{C}_{\mathbf{S}}) \cap De(\mathbf{S})$, then $\exists S \in \mathbf{S}$ such that $S \in An(\{W\}) \subseteq An(\mathbf{C}_{\mathbf{S}})$ contradicts $\mathbf{C}_{\mathbf{S}} \cap De(\mathbf{S}) = \emptyset$). Therefore, we have $\mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{C}_{\mathbf{S}} \setminus De(\mathbf{S})) \setminus De(\mathbf{S})}} = \mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{C}_{\mathbf{S}})}}$; Let $\mathbf{A} = \mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{C}_{\mathbf{S}} \setminus De(\mathbf{S})) \setminus De(\mathbf{S})}}$. We now show that $\mathbf{C}_{\mathbf{S}} \supseteq \mathbf{A}$. Consider some variable $W \in \mathbf{A}$. Then, there exists a variable $Y \in \mathbf{C}_{\mathbf{S}} \setminus De(\mathbf{S})$ such that $W \in An(\{Y\})$. Moreover, since $Y \in \mathbf{C}_{\mathbf{S}} \subseteq \mathbf{A}$, we either have $W = Y$ (and hence $W \in \mathbf{C}_{\mathbf{S}}$) or a path $W = V_k \leftrightarrow \cdots \leftrightarrow V_1 \leftrightarrow Y$ for some $k \geq 1$ with $V_i \in An(\mathbf{C}_{\mathbf{S}} \setminus De(\mathbf{S})) \setminus De(\mathbf{S})$ for each $i \in [k]$ (by the construction of $\mathbf{A}$). We show by induction that $V_i \in \mathbf{C}_{\mathbf{S}}$ for each $i \in [k]$.

Base case. $k = 1$. Since $V_1 \in An(\mathbf{C}_{\mathbf{S}} \setminus De(\mathbf{S})) \setminus De(\mathbf{S})$, we have $\{V_1\} \cap De(\mathbf{S}) = \emptyset$. Moreover, $V_1 \in An(\mathbf{C}_{\mathbf{S}}) \subseteq An(\mathbf{C})$ (since $\mathbf{C}_{\mathbf{S}} \subseteq \mathbf{C}$). Furthermore, $Y \in \mathbf{C}_{\mathbf{S}} \subseteq \mathbf{C}$. So, $Y \in \mathbf{C}$, $V_1 \in An(\mathbf{C})$, $V_1 \leftrightarrow Y$, and $\mathbf{C}$ is an AC by assumption implies that $V_1 \in \mathbf{C}$. Therefore, $V_1 \in \mathbf{C} \setminus De(\mathbf{S})$. Since $Y \in \mathbf{C}_{\mathbf{S}} = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{C} \setminus De(\mathbf{S})}}$, $Y \leftrightarrow V_1$, and $V_1 \in \mathbf{C} \setminus De(\mathbf{S})$, we get $V_1 \in \mathbf{C}_{\mathbf{S}}$.

Inductive assumption. Assume, for some $i \in [k-1]$, we have $V_i \in \mathbf{C}_{\mathbf{S}}$.

Inductive step. By similar reasoning as in the base case, we show that $V_{i+1} \in \mathbf{C}_{\mathbf{S}}$. Since $V_{i+1} \in An(\mathbf{C}_{\mathbf{S}} \setminus De(\mathbf{S})) \setminus De(\mathbf{S})$, we have $\{V_{i+1}\} \cap De(\mathbf{S}) = \emptyset$. Moreover, $V_{i+1} \in An(\mathbf{C}_{\mathbf{S}}) \subseteq An(\mathbf{C})$. Furthermore, by the inductive assumption, $V_i \in \mathbf{C}_{\mathbf{S}} \subseteq \mathbf{C}$. So, $V_i \in \mathbf{C}$, $V_{i+1} \in An(\mathbf{C})$, $V_{i+1} \leftrightarrow V_i$, and $\mathbf{C}$ being an AC implies that $V_{i+1} \in \mathbf{C}$. Therefore, $V_{i+1} \in \mathbf{C} \setminus De(\mathbf{S})$. Since $V_i \in \mathbf{C}_{\mathbf{S}}$, $V_{i+1} \leftrightarrow V_i$, and $V_{i+1} \in \mathbf{C} \setminus De(\mathbf{S})$, we get $V_{i+1} \in \mathbf{C}_{\mathbf{S}}$.

Therefore, $W \in \mathbf{C}_{\mathbf{S}}$ and since W was chosen arbitrarily from $\mathbf{A}$, we have $\mathbf{A} \subseteq \mathbf{C}_{\mathbf{S}}$. □

Lemma 2 (Correctness of LISTCIX). LISTCIX $(\mathcal{G}_{\mathbf{V} \leq X}, X, \mathbf{V}^{\leq X}, \mathbf{I}, \mathbf{R})$ enumerates all and only all non-vacuous conditional independence relations invoked by the c-component local Markov property associated with X and admissible ancestral c-components $\mathbf{C}$ relative to X where $\mathbf{I} \subseteq \mathbf{C} \subseteq \mathbf{R}$. Further, LISTCIX runs in $O(n^2(n+m))$

delay where n and m represent the number of nodes and edges in $\mathcal{G}$, respectively.

Proof. We show the correctness of ListCIX and the running time that ListCIX runs in $O(n^2(n+m))$ delay.

- Correctness: We prove correctness by structural induction on the binary recursion tree for ListCIX, rooted at $\mathcal{N}(\mathbf{I}, \mathbf{R})$. We claim that ListCIX called at a node $\mathcal{N}(\mathbf{I}', \mathbf{R}')$ enumerates all and only non-vacuous CIs of X invoked by C-LMP (Def. 5) such that the conditioning set is of the form $Pa(\mathbf{C}) \setminus \{X\}$ for some AC $\mathbf{C}$ such that $\mathbf{I}' \subseteq \mathbf{C} \subseteq \mathbf{R}'$.
  Base case. Consider a leaf node $\mathcal{N}(\mathbf{I}', \mathbf{R}')$ [7]. Let

  $$\mathbf{C} := \textsc{FindAAC}(\mathcal{G}_{\mathbf{V}^{\leq X}}, X, \mathbf{V}^{\leq X}, \mathbf{I}', \mathbf{R}').$$

  Since we are at a leaf node, we either have
  1. $\mathbf{C} = \perp$, in which case ListCIX outputs nothing at $\mathcal{N}(\mathbf{I}', \mathbf{R}')$. By the correctness of FindAAC (Lemma 1) there are no CIs of X invoked by C-LMP such that the conditioning set is of the form $Pa(\mathbf{C}) \setminus \{X\}$ for some AC $\mathbf{C}$ such that $\mathbf{I}' \subseteq \mathbf{C} \subseteq \mathbf{R}'$. Therefore, the output is correct.
  2. $\mathbf{I}' = \mathbf{R}'$ and hence $\mathbf{C} = \mathbf{I}'$. Similarly, by the correctness of FindAAC and the definition of C-LMP, ListCIX outputs the unique non-vacuous CI of the desired form at $\mathcal{N}(\mathbf{I}', \mathbf{R}')$.

  Note that these are the only conditions under which we are at a leaf node. If $\mathbf{C} \neq \perp$ and $\mathbf{I}' \neq \mathbf{R}'$, then $\mathbf{T} = \mathbf{R}' \cap (Sp(\mathbf{I}') \setminus \mathbf{I}')$ must be non-empty (because $\mathbf{R}'$ is a c-component such that $\mathbf{R}' \supsetneq \mathbf{I}'$) and we recurse.
  Inductive assumption. Assume the claim holds for some nodes $\mathcal{N}_1(\mathbf{I}_1, \mathbf{R}_1), \mathcal{N}_2(\mathbf{I}_2, \mathbf{R}_2)$.
  Inductive step. We show that the claim holds for any node $\mathcal{N}_0(\mathbf{I}', \mathbf{R}')$ whose children are $\mathcal{N}_1(\mathbf{I}_1, \mathbf{R}_1), \mathcal{N}_2(\mathbf{I}_2, \mathbf{R}_2)$. We first define three collections of ACs relative to X. Recall that $\mathcal{AC}_X$ (Def. A.1.4) denotes the set of all ACs relative to X.

  $$\mathcal{AC}_0 = \{\mathbf{C} \subseteq \mathbf{V}^{\leq X} \mid \mathbf{C} \in \mathcal{AC}_X, \mathbf{C} \text{ is AAC}, \mathbf{I}' \subseteq \mathbf{C} \subseteq \mathbf{R}'\}$$

  $$\mathcal{AC}_1 = \{\mathbf{C} \subseteq \mathbf{V}^{\leq X} \mid \mathbf{C} \in \mathcal{AC}_X, \mathbf{C} \text{ is AAC}, \mathbf{I}_1 \subseteq \mathbf{C} \subseteq \mathbf{R}_1\}$$

  $$\mathcal{AC}_2 = \{\mathbf{C} \subseteq \mathbf{V}^{\leq X} \mid \mathbf{C} \in \mathcal{AC}_X, \mathbf{C} \text{ is AAC}, \mathbf{I}_2 \subseteq \mathbf{C} \subseteq \mathbf{R}_2\}$$

  It suffices to show that $\mathcal{AC}_0 = \mathcal{AC}_1 \cup \mathcal{AC}_2$, since this implies that any CI that should be output by ListCIX at $\mathcal{N}_0(\mathbf{I}', \mathbf{R}')$ is output by ListCIX at either $\mathcal{N}_1(\mathbf{I}_1, \mathbf{R}_1)$ or $\mathcal{N}_2(\mathbf{I}_2, \mathbf{R}_2)$. Since ListCIX at $\mathcal{N}_0(\mathbf{I}', \mathbf{R}')$ calls ListCIX at both $\mathcal{N}_1(\mathbf{I}_1, \mathbf{R}_1)$ and $\mathcal{N}_2(\mathbf{I}_2, \mathbf{R}_2)$, we prove the claim.
  By construction at lines 10-11, since $\mathcal{N}_0(\mathbf{I}', \mathbf{R}')$ has children $\mathcal{N}_1(\mathbf{I}_1, \mathbf{R}_1)$ and $\mathcal{N}_2(\mathbf{I}_2, \mathbf{R}_2)$, we have (without loss of generality) that $(\mathbf{I}_1, \mathbf{R}_1) = (\mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{I}' \cup \{S\})}}, \mathbf{R}')$ and $(\mathbf{I}_2, \mathbf{R}_2) = (\mathbf{I}', \mathcal{C}(X)_{\mathcal{G}_{\mathbf{R}' \setminus De(\{S\})}})$ for some $S \in \mathbf{T} = \mathbf{R}' \cap (Sp(\mathbf{I}') \setminus \mathbf{I}')$.
  First, some technicalities. we want to show that $(\mathbf{I}_1, \mathbf{R}_1), (\mathbf{I}_2, \mathbf{R}_2)$ are well-defined and non-vacuous inputs to ListCIX.
  - Since $S \in \mathbf{T} \subseteq \mathbf{R}'$ and $\mathbf{I}' \subseteq \mathbf{R}'$, and $\mathbf{R}'$ is an AC relative to X, we have $\mathbf{I}_1 = \mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{I}' \cup \{S\})}} \subseteq \mathbf{R}' = \mathbf{R}_1$. Furthermore, $\mathbf{I}_1$ is an AC by construction.
  - Since $S \in Sp(\mathbf{I}') \setminus \mathbf{I}'$ and $\mathbf{I}'$ is an AC, we have $De(\{S\}) \cap \mathbf{I}' = \emptyset$. Since $\mathbf{I}' \subseteq \mathbf{R}'$, this further implies that $\mathbf{I}_2 = \mathbf{I}' \subseteq \mathbf{R}_2 = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{R}' \setminus De(\{S\})}}$. Moreover, $\mathbf{R}_2$ is an AC by Prop. C.2.3.

  We show the equality of $\mathcal{AC}_0$ and $\mathcal{AC}_1 \cup \mathcal{AC}_2$ in both directions.
  - $\mathcal{AC}_1 \cup \mathcal{AC}_2 \subseteq \mathcal{AC}_0$
    For any $\mathbf{C} \in \mathcal{AC}_1 \cup \mathcal{AC}_2$, we have either
    * $\mathbf{I}_1 \subseteq \mathbf{C} \subseteq \mathbf{R}_1$ and hence $\mathbf{I}' \subseteq \mathbf{C} \subseteq \mathbf{R}'$ since $\mathbf{I}' \subseteq \mathbf{I}_1, \mathbf{R}' = \mathbf{R}_1$, *or*
    * $\mathbf{I}_2 \subseteq \mathbf{C} \subseteq \mathbf{R}_2$ and hence $\mathbf{I}' \subseteq \mathbf{C} \subseteq \mathbf{R}'$ since $\mathbf{I}' = \mathbf{I}_2, \mathbf{R}_2 \subseteq \mathbf{R}'$.

    Therefore, $\mathcal{AC}_1 \cup \mathcal{AC}_2 \subseteq \mathcal{AC}_0$.
  - $\mathcal{AC}_1 \cup \mathcal{AC}_2 \supseteq \mathcal{AC}_0$
    For any $\mathbf{C} \in \mathcal{AC}_0$, we have $\mathbf{I}' \subseteq \mathbf{C} \subseteq \mathbf{R}'$. Then, we have either
    * $S \in \mathbf{C}$. Therefore, $\mathbf{I}' \cup \{S\} \subseteq \mathbf{C}$. Since $\mathbf{C}$ is an AC, we must have $\mathcal{C}(X)_{\mathcal{G}_{An(\mathbf{I}' \cup \{S\})}} \subseteq \mathbf{C}$ (otherwise, there is an ancestor of $\mathbf{I}' \cup \{S\}$ not in $\mathbf{C}$ which is connected by a bi-directed path to some node in $\mathbf{I} \cup \{S\} \subseteq \mathbf{C}$). Therefore, we have $\mathbf{I}_1 \subseteq \mathbf{C} \subseteq \mathbf{R}_1 = \mathbf{R}'$ and hence $\mathbf{C} \in \mathcal{AC}_1$.
    * $S \notin \mathbf{C}$. Since $\mathbf{C}$ is an AC, this implies $De(\{S\}) \cap \mathbf{C} = \emptyset$. Moreover, $\mathbf{C} \subseteq \mathbf{R}'$ further implies that $\mathbf{C} \subseteq \mathcal{C}(X)_{\mathcal{G}_{\mathbf{R}' \setminus De(\{S\})}}$. Therefore, we have $\mathbf{I}_2 = \mathbf{I}_1 \subseteq \mathbf{C} \subseteq \mathbf{R}_2$, and $\mathbf{C} \in \mathcal{AC}_2$.

    Therefore, $\mathcal{AC}_1 \cup \mathcal{AC}_2 \supseteq \mathcal{AC}_0$.

  Hence, $\mathcal{AC}_0 = \mathcal{AC}_1 \cup \mathcal{AC}_2$ and we are done. Since ListCIX enumerates all AACs relative to X correctly, by Thm. 2, it enumerates all non-vacuous CIs invoked by C-LMP for X correctly.
- Running time:
  Consider the recursion tree for ListCIX. Whenever a tree node $\mathcal{N}(\mathbf{I}', \mathbf{R}')$ is visited, the function FindAAC is called, which takes $O(n(n+m))$ time (by Lemma 1). If FindAAC outputs $\perp$, then ListCIX does not search further from $\mathcal{N}$ because there exists no AAC $\mathbf{C}$ relative to X with $\mathbf{I}' \subseteq \mathbf{C} \subseteq \mathbf{R}'$. Otherwise, recursion continues until a leaf tree node is visited. In each level of the tree, a single node S is removed from $\mathbf{T}$. Either the variable S, is added to $\mathbf{I}$, resulting in $\mathbf{I}'$ given at line 9 (by $S \in Sp(\mathbf{I}) \setminus \mathbf{I}$), or the variable S is removed from $\mathbf{R}$ to construct $\mathbf{R}'$ which is shown at line 9. Any $\mathbf{C}$ is uniquely contained in one child; therefore, no CI is output more than once. The depth of the tree is at most n, and the time required to find one $\mathbf{C}$ and output one non-vacuous CI invoked by C-LMP associated with $\mathbf{C}$ (following Def. 5), is $O(n^2(n+m))$. In the worst case scenario, n branches will be aborted (i.e.,

[7] A leaf node is a node that has no children.

FINDAAC outputs $\perp$ on every level of the tree) before reaching the first leaf. It takes $O(n^2(n+m))$ time to produce either the first output or halt.

□

Theorem 3 (Correctness of LISTCI). Let $\mathcal{G}$ be a causal graph and $\mathbf{V}^{\prec}$ a consistent ordering. LISTCI($\mathcal{G}, \mathbf{V}^{\prec}$) enumerates all and only all non-vacuous conditional independence relations invoked by the c-component local Markov property in $O(n^2(n+m))$ delay where n and m represent the number of nodes and edges in $\mathcal{G}$, respectively.

Proof. We show the correctness of LISTCI and the running time that LISTCI runs in $O(n^2(n+m))$ delay.

- Correctness:
  LISTCI($\mathcal{G}, \mathbf{V}^{\prec}$) iterates over each variable $X \in \mathbf{V}^{\prec}$. For each X, LISTCI constructs two ACs relative to X: $\mathbf{I} = \mathcal{C}(X)_{\mathcal{G}_{An(\{X\})}}$ and $\mathbf{R} = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{V}\leq X}}$. $\mathbf{I}$ is the minimum-size AC relative to X and $\mathbf{R}$ is the maximum-size AC relative to X. Then, LISTCI calls the function LISTCIX($\mathcal{G}_{\mathbf{V}\leq X}, X, \mathbf{V}^{\leq X}, \mathbf{I}, \mathbf{R}$) that outputs all and only all non-vacuous CIs invoked by C-LMP associated with X, which is performed by generating all and only all AACs $\mathbf{C}$ relative to X under the constraint $\mathbf{I} \subseteq \mathbf{C} \subseteq \mathbf{R}$ (by Lemma 2). LISTCI($\mathcal{G}, \mathbf{V}^{\prec}$) iterates over each variable $X \in \mathbf{V}^{\prec}$, and thus LISTCI($\mathcal{G}, \mathbf{V}^{\prec}$) lists all and only all non-vacuous CIs invoked by C-LMP.
- Running time:
  There are two types of worst case scenarios.
  1. No CI is invoked by C-LMP.
     By the assumption that no CI is invoked by C-LMP, Def. 5 and Def. 6, none of the ACs $\mathbf{C}$ relative to X for any $X \in \mathbf{V}^{\prec}$ is admissible. For each X visited by LISTCI($\mathcal{G}, \mathbf{V}^{\prec}$), LISTCI calls the function LISTCIX($\mathcal{G}_{\mathbf{V}\leq X}, X, \mathbf{V}^{\leq X}, \mathbf{I}, \mathbf{R}$) at line 5. LISTCIX spends $O(n(n+m))$ time and terminates with no output since LISTCIX calls the function FINDAAC($\mathcal{G}_{\mathbf{V}\leq X}, X, \mathbf{V}^{\leq X}, \mathbf{I}, \mathbf{R}$) at line 3, which returns $\perp$ (by Lemma 1). LISTCI checks the next variable of X, if any exists. Since $|\mathbf{V}^{\prec}| = n$, LISTCI spends $O(n^2(n+m))$ time and then terminates with no output.
  2. No CI invoked by C-LMP exists for all variables $X \in \mathbf{V}^{\prec}$ except for the last variable X_n in the ordering $\mathbf{V}^{\prec}$. For the first $n-1$ variables in $\mathbf{V}^{\prec}$, LISTCI spends $O(n^2(n+m)) = O(n(n+m) * (n-1))$ time producing no output. More specifically, for each variable, LISTCIX spends $O(n(n+m))$ time and terminates with no output since LISTCIX calls FINDAAC($\mathcal{G}_{\mathbf{V}\leq X}, X, \mathbf{V}^{\leq X}, \mathbf{I}, \mathbf{R}$) at line 3, which returns $\perp$. When $X = X_n$, LISTCI calls the function LISTCIX($\mathcal{G}_{\mathbf{V}\leq X}, X, \mathbf{V}^{\leq X}, \mathbf{I}, \mathbf{R}$) at line 5 where LISTCIX spends $O(n^2(n+m))$ time to output one non-vacuous CI invoked by C-LMP that is associated with X_n. In total, LISTCI spends $O(n^2(n+m))$ time to produce an output.

□

C.3 Appendix Proofs

Proposition C.3.1 (Total Number of CIs Invoked by GMP)**.** *Given a causal graph $\mathcal{G}$ over a set of variables $\mathbf{V}$ with $n = |\mathbf{V}|$, the global Markov property for $\mathcal{G}$ invokes $O(4^n)$ number of conditional independence relations. Moreover, there exists a causal graph $\mathcal{G}$ for which the bound is tight, that is, the global Markov property for $\mathcal{G}$ implies $\Omega(4^n)$ number of conditional independence relations.*

Proof. Each CI invoked by GMP is given by a choice of disjoint sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ where $\mathbf{X}, \mathbf{Y} \neq \emptyset$ and a d-separation statement $\mathbf{X} \perp_d \mathbf{Y} \mid \mathbf{Z}$. The number of such statements is upper-bounded by

$$d(n) = \frac{1}{2}\sum_{i=1}^{n}\binom{n}{i}\sum_{j=1}^{n-i}\binom{n-i}{j}\sum_{k=0}^{n-i-j}\binom{n-i-j}{k} \tag{22}$$

$$= \frac{2^n(2^n-1)}{2} - 3^n + 2^n \tag{23}$$

where we divide the quantity by 2 to avoid double-counting the following two symmetrical statements: $\mathbf{X} \perp_d \mathbf{Y} \mid \mathbf{Z}$ and $\mathbf{Y} \perp_d \mathbf{X} \mid \mathbf{Z}$ (since d-separation is symmetric). We first simplify the inner-most sum.

$$\sum_{k=0}^{n-i-j}\binom{n-i-j}{k} = 2^{n-i-j} \tag{24}$$

This gives

$$d(n) = \frac{1}{2}\sum_{i=1}^{n}\binom{n}{i}\sum_{j=1}^{n-i}\binom{n-i}{j}2^{n-i-j} \tag{25}$$

We then simplify the second nested sum. Note that by the binomial theorem,

$$3^{n-i} = (2+1)^{n-i} \tag{26}$$

$$= \sum_{j=0}^{n-i}\binom{n-i}{j}2^{n-i-j}1^j \tag{27}$$

$$= 2^{n-i} + \sum_{j=1}^{n-i}\binom{n-i}{j}1^j 2^{n-i-j} \tag{28}$$

Therefore,

$$d(n) = \frac{1}{2}\sum_{i=1}^{n}\binom{n}{i}(3^{n-i} - 2^{n-i}) \tag{29}$$

$$= \frac{1}{2}\left(\sum_{i=1}^{n}\binom{n}{i}3^{n-i} - \sum_{i=1}^{n}\binom{n}{i}2^{n-i}\right) \tag{30}$$

$$= \frac{1}{2}(4^n + 2^n) - 3^n \in O(4^n). \tag{31}$$

Moreover, for $\mathcal{G}$ given by the independent set on n variables (i.e., $\mathcal{G}$ contains no edges), every possible d-separation holds, therefore we get $d(n) \in \Omega(4^n)$ number of CIs implied by GMP. □

D Discussion and Examples

D.1 Explaining Markov properties

A causal DAG on n variables may encode $\Theta(4^n)$ CIs (Prop. C.3.1). In this section, we explain how a subset of these CIs, often considered a 'basis' (Bareinboim et al. 2022), may imply all others.

The CI relation is a *semi-graphoid* (Pearl 1986, 1988). Given an arbitrary probability distribution $P(\mathbf{v})$ over a set of variables $\mathbf{V}$, CIs in $P(\mathbf{v})$ must exhibit certain properties. Specifically, for disjoint sets of variables $\mathbf{X}, \mathbf{Y}, \mathbf{W}, \mathbf{Z}$, where $\mathbf{X}, \mathbf{Y} \neq \emptyset$, the probability axioms can be used to show that the following properties hold:

1. Symmetry: $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \iff \mathbf{Y} \perp\!\!\!\perp \mathbf{X} \mid \mathbf{Z}$
2. Decomposition: $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z} \implies \mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$ and $\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z}$
3. Weak union:$\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z} \implies \mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \mathbf{W}$ and $\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z} \cup \mathbf{Y}$
4. Contraction: $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$ and $\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z} \cup \mathbf{Y} \implies \mathbf{X} \perp\!\!\!\perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z}$

We give an example to show how these axioms can be applied.

■ **Example D.1.1.** The DAG $\mathcal{G}$ (Fig. D.2.1a) encodes 5 CIs, but the colored subsets of CIs can be used to derive all others, as shown in Fig. D.2.1b. For one example, the CI $X_4 \perp\!\!\!\perp \{X_1, X_2\}$ implies all others. In the context of testing $\mathcal{G}$ against observational data, it suffices to test only $X_4 \perp\!\!\!\perp \{X_1, X_2\}$. For another example, $X_4 \perp\!\!\!\perp X_2$ and $X_1 \perp\!\!\!\perp X_4 \mid X_2$ together imply $X_4 \perp\!\!\!\perp \{X_1, X_2\}$ by the contraction axiom, and hence all other CIs. Therefore, it suffices also to only test $X_4 \perp\!\!\!\perp X_2$ and $X_1 \perp\!\!\!\perp X_4 \mid X_2$. □

In the given example, scrutiny revealed which CIs are sufficient to derive others via the semi-graphoid axioms. Markov properties, however, provide a systematic way to identify such CIs. The semi-graphoid axioms can be used to show equivalence between Markov properties: for example, between the local Markov property and GMP for Markovian DAGs (Pearl 1988; Lauritzen et al. 1990; Lauritzen 1996), and between (LMP,$\prec$) and GMP for semi-Markovian DAGs (Richardson 2003).

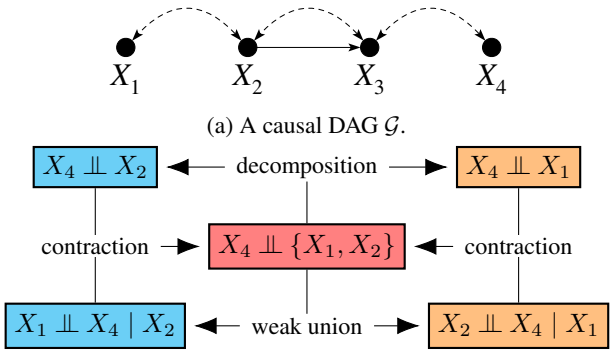
D.2 C-LMP and the Semi-Markov Factorisation

In this section, we develop a connection between Markov properties and a related notion of compatibility between causal graphs and observational data – the factorisation that the distribution should admit. This offers another perspective on the combinatorial explosion in the number of CIs invoked by C-LMP in the semi-Markovian case, compared with the Markovian case.

An observational distribution $P(\mathbf{v})$ over a set of variables $\mathbf{V}$ factorizes, according to the chain rule, as

$$P(\mathbf{v}) = \prod_{V_i \in \mathbf{V}} p(v_i \mid v_1, \ldots, v_{i-1})$$

However, if we know $P(\mathbf{v})$ is compatible with a graph $\mathcal{G}$, CIs implied by $\mathcal{G}$ can be used to simplify the factorisation above to a *Markov factorisation*. We first define the Markov factorisation for Markovian DAGs.



(a) A causal DAG $\mathcal{G}$.

(b) CIs encoded in $\mathcal{G}$.

Figure D.2.1: A causal DAG $\mathcal{G}$ and a hyper-graph depicting all the CIs encoded in $\mathcal{G}$. An edge indicates that we can derive the CIs at the arrowheads from the CIs at the tails using the semi-graphoid axioms. The highlighted subsets of CIs in blue, orange, and pink are each sufficient to derive all other CIs in the graph.

Definition D.2.1 (Markov Relative (Bareinboim et al. 2022))**.** An observational distribution $P(\mathbf{v})$ is said to be Markov relative to a graph $\mathcal{G}$ (over a set of variables $\mathbf{V}$) if, for a given ordering $V_1, \ldots, V_n$ consistent with $\mathcal{G}$, $P(\mathbf{v})$ factorizes as

$$P(\mathbf{v}) = \prod_{V_i \in \mathbf{V}} p(v_i \mid pa_i^-) \tag{32}$$

where $Pa_i^- = Pa(\{V_i\})_{\mathcal{G}} \setminus \{V_i\}$.

For example, in the simple three-node graph $X \to Y \to Z$, the observational distribution $P(x, y, z)$ factorizes as

$$P(x)P(y \mid x)P(z \mid y, x) = P(x)P(y \mid x)P(z \mid y)$$

By means of this factorisation, a graph imposes CI constraints on the distribution $P(\mathbf{v})$: in our example, $Z \perp\!\!\!\perp X \mid Y$. This gives an equivalent definition of 'Markov relative': $P(\mathbf{v})$ is Markov relative to $\mathcal{G}$ if, for a given ordering $\mathbf{V}^{\prec}$ consistent with $\mathcal{G}$, and for each $V_i \in \mathbf{V}$,

$$V_i \perp\!\!\!\perp \mathbf{V}^{\leq V_i} \setminus Pa(\{V_i\}) \mid Pa(\{V_i\}) \setminus \{V_i\}$$

Notice how this set of CI constraints is identical to C-LMP in the Markovian case (Eq. (4)), discussed in Section 3. Therefore, if $P(\mathbf{v})$ is Markov relative to a given $\mathcal{G}$, all the CI constraints encoded in $\mathcal{G}$ must hold in $P(\mathbf{v})$.

If $\mathcal{G}$ contains bidirected edges, the factorisation in Def. D.2.1 no longer applies. For instance, a variable V_i may be connected to a non-descendant V_j by means of a bidirected edge, V_i is not independent of V_j when conditioning on the (observed) parents Pa_i^-. This leads to the more general definition of compatibility for semi-Markovian graphs, given below.

Definition D.2.2 (Semi-Markov Relative (Bareinboim et al. 2022))**.** An observational distribution $P(\mathbf{v})$ is said to be Semi-Markov relative to a graph $\mathcal{G}$ (over a set of variables

$\mathbf{V}$) if, for every ordering $V_1, \ldots, V_n$ consistent with $\mathcal{G}$, $P(\mathbf{v})$ factorizes as

$$P(\mathbf{v}) = \prod_{V_i \in \mathbf{V}} p(v_i \mid pa_i^+) \tag{33}$$

where $Pa_i^+ = Pa\Big(\mathcal{C}(V_i)_{\mathcal{G}_{\mathbf{V}^{\leq V_i}}}\Big)_{\mathcal{G}} \setminus \{V_i\}$.

■ **Example D.2.1.** Consider the semi-Markovian graph $\mathcal{G}$ in Fig. D.2.1a. There are 12 possible orderings of the 4 nodes; each ordering induces a factorisation of $P(\mathbf{v})$ in Def. D.2.2. We give four examples.

1. $X_1 \prec X_2 \prec X_3 \prec X_4$. This implies $Pa^+(\{X_1\}) = \emptyset$, $Pa^+(\{X_2\}) = \{X_1\}$, $Pa^+(\{X_3\}) = \{X_1, X_2\}$, $Pa^+(\{X_4\}) = \{X_1, X_2, X_3\}$. The resultant semi-Markov factorisation is $p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2)p(x_4 \mid x_1, x_2, x_3)$. This is equivalent to the factorisation given by the chain rule, and implies no CI constraints.
2. $X_1 \prec X_2 \prec X_4 \prec X_3$. This implies $Pa^+(\{X_1\}) = \emptyset$, $Pa^+(\{X_2\}) = \{X_1\}$, $Pa^+(\{X_3\}) = \{X_1, X_2, X_4\}$, $Pa^+(\{X_4\}) = \emptyset$. The resultant semi-Markov factorisation is $p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2 \mid x_1)p(x_4)p(x_3 \mid x_1, x_2, x_4)$, which implies the CI constraint $X_4 \perp\!\!\!\perp \{X_1, X_2\}$.
3. $X_2 \prec X_4 \prec X_1 \prec X_3$. This implies $Pa^+(\{X_2\}) = \emptyset$, $Pa^+(\{X_4\}) = \emptyset$, $Pa^+(\{X_1\}) = \{X_2\}$, $Pa^+(\{X_3\}) = \{X_1, X_2, X_4\}$. The resultant semi-Markov factorisation is $p(x_1, x_2, x_3, x_4) = p(x_2)p(x_4)p(x_1 \mid x_2)p(x_3 \mid x_1, x_2, x_4)$, which implies the CI constraints $X_4 \perp\!\!\!\perp \{X_2\}$ and $X_1 \perp\!\!\!\perp \{X_4\} \mid \{X_2\}$.
4. $X_4 \prec X_1 \prec X_2 \prec X_3$. This implies $Pa^+(\{X_4\}) = \emptyset$, $Pa^+(\{X_1\}) = \emptyset$, $Pa^+(\{X_2\}) = \{X_1\}$, $Pa^+(\{X_3\}) = \{X_1, X_2, X_4\}$. The resultant semi-Markov factorisation is $p(x_1, x_2, x_3, x_4) = p(x_4)p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2, x_4)$, which implies the CI constraints $X_1 \perp\!\!\!\perp \{X_4\}$ and $X_2 \perp\!\!\!\perp \{X_4\} \mid \{X_1\}$.

The CIs induced by the first ordering (namely, none) are clearly insufficient to derive all CIs encoded in $\mathcal{G}$, similar to Ex. 5. The four orderings above are chosen to be representative. Each of the eight remaining orderings induces exactly the same CIs as one the four orderings. □

We define the exact set of CI constraints implied by the semi-Markov factorisation below.

Definition D.2.3 (Semi-Markov Relative CI Constraints)**.** Let $\mathcal{G}$ be a causal graph over variables $\mathbf{V}$ and $P(\mathbf{v})$ a probability distribution over $\mathbf{V}$ that is semi-Markov relative to $\mathcal{G}$. Then, the conditional independence constraints encoded in the factorisation of $P(\mathbf{v})$ are given by: For every ordering $\mathbf{V}^{\prec}$ consistent with $\mathcal{G}$, for every variable $V_i \in \mathbf{V}$,

$$V_i \perp\!\!\!\perp \mathbf{V}^{\leq V_i} \setminus (Pa^+(\{V_i\}) \cup \{V_i\}) \mid Pa^+(\{V_i\})$$

where $Pa^+(\{V_i\}) = Pa\Big(\mathcal{C}(V_i)_{\mathcal{G}_{\mathbf{V}^{\leq V_i}}}\Big)_{\mathcal{G}_{\mathbf{V}^{\leq V_i}}} \setminus \{V_i\}$ and $\mathbf{V}^{\leq V_i}$ depends on the ordering $\mathbf{V}^{\prec}$.

Note that we take a union over all orderings in Def. D.2.3.

■ **Example D.2.2.** Continuing Ex. D.2.1. The set of CIs induced by the semi-Markov factorization for $\mathcal{G}$ (Fig. D.2.1a) is the union of all CIs listed in Ex.D.2.1: $X_4 \perp\!\!\!\perp \{X_1, X_2\}, X_4 \perp\!\!\!\perp \{X_2\}, X_1 \perp\!\!\!\perp \{X_4\} \mid \{X_2\}, X_1 \perp\!\!\!\perp \{X_4\}, X_2 \perp\!\!\!\perp \{X_4\} \mid \{X_1\}$. □

Ex. D.2.1 and Ex. D.2.2 show an important contrast between the Markovian and semi-Markovian cases. In the Markovian case, we can fix an arbitrary ordering: compatibility requires that $P(\mathbf{v})$ factorize according to the product in Def. D.2.1 for any one ordering. In the semi-Markovian case, we cannot fix an arbitrary ordering; the ordering $X_1 \prec X_2 \prec X_3 \prec X_4$ in Ex. D.2.1 provides no CI constraints. Coincidentally, the ordering $X_1 \prec X_2 \prec X_4 \prec X_3$ does suffice to derive all CIs encoded in $\mathcal{G}$ from $X_4 \perp\!\!\!\perp \{X_1, X_2\}$, as seen in Fig. D.2.1b. However, no method is known for choosing an ordering (or subset of orderings) that a priori guarantees that all CIs encoded in the graph can be derived from the resulting factorisation(s). Therefore, in the semi-Markovian case, it is required that $P(\mathbf{v})$ factorizes according to the product in Def. D.2.2 for all possible orderings.

Applying Def. D.2.3 to a Markovian graph $\mathcal{G}$ reveals why considering all orderings is not necessary in the Markovian case. We make two observations for Markovian $\mathcal{G}$:

1. Each c-component in $\mathcal{G}$ is a singleton. This means $\mathcal{C}(V_i)_{\mathcal{G}_{\mathbf{V}^{\leq V_i}}} = \{V_i\}$.
2. The parents of a variable precede it in every ordering, and do not depend on the ordering. This means $Pa(\{V_i\})_{\mathcal{G}_{\mathbf{V}^{\leq V_i}}} = Pa(\{V_i\})_{\mathcal{G}}$.

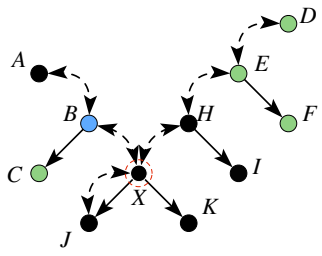
Therefore, for any ordering $\mathbf{V}^{\prec}$ and any variable $V_i \in \mathbf{V}$, the set $Pa^+(\{V_i\})$ simplifies to $Pa(\{V_i\})_{\mathcal{G}} \setminus \{V_i\}$. The set of CIs induced by Def. D.2.3 contains: for every ordering $\mathbf{V}^{\prec}$ consistent with $\mathcal{G}$, for every variable $V_i \in \mathbf{V}$,

$$V_i \perp\!\!\!\perp \mathbf{V}^{\leq V_i} \setminus Pa(\{V_i\})_{\mathcal{G}} \mid Pa(\{V_i\})_{\mathcal{G}} \setminus \{V_i\} \tag{34}$$

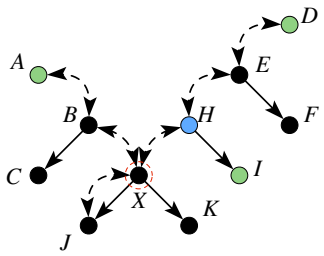
Let Φ denote this set of CIs. The set of CIs induced by the Markov factorisation for $\mathcal{G}$ (Def. D.2.1) – which fixes one ordering – is a subset of Φ. Moreover, contrast Φ with the set of CIs induced by LMP. LMP abstracts away the ordering of variables. Since $Nd(\{V_i\}) = \bigcup_{\prec \text{ ordering of } \mathcal{G}} \mathbf{V}_{\prec}^{\leq V_i} \setminus \{V_i\}$, LMP tests the CI: $V_i \perp\!\!\!\perp Nd(\{V_i\}) \setminus Pa(\{V_i\})_{\mathcal{G}} \mid (Pa(\{V_i\})_{\mathcal{G}} \setminus \{V_i\})$. This CI implies the CI in Eq. (34) for every possible ordering by the decomposition axiom.

Therefore, we have another perspective on the combinatorial explosion of the number of CIs in the semi-Markovian case, relative to the Markovian case. This explosion was introduced in Section 3, and characterised in terms of ACs. Here, we understand it through the many possible orderings of a given graph. To tie together these two concepts, we show an equivalence between C-LMP and the CI constraints invoked by the semi-Markov factorization.

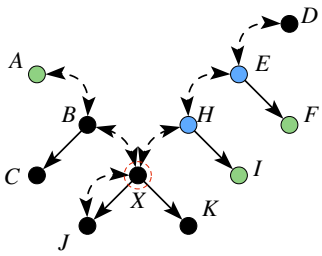
Proposition D.2.1. *Given a causal graph $\mathcal{G}$ over a set of variables $\mathbf{V}$ and a consistent ordering $\mathbf{V}^{\prec}$, let $\mathcal{L}^C$ denote the set of conditional independence constraints that the c-component local Markov property invokes for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$ and $\mathcal{L}^P$ denote the set of conditional independence constraints that the semi-Markov factorisation induces for*



(a) X is separated from C but not A when conditioning on B.



(b) X is separated from A but not C when not conditioning on B.



(c) X is separated from F, I but not D when conditioning on H, E.

Figure D.2.2: Fig. 2 reproduced for convenience. Three ACs relative to the variable X in the (same) causal DAG $\mathcal{G}$. Assume an ordering $A \prec B \prec \cdots \prec X \prec J \prec K$. The ACs relative to X (excluding $\{X\}$ itself), shown in blue, separate it from the variables shown in green.

$\mathcal{G}$. Then, $\mathcal{L}^C \subseteq \mathcal{L}^P$. Moreover, there exists $\mathcal{G}, \mathbf{V}^\prec$ for which $\mathcal{L}^C \subsetneq \mathcal{L}^P$.

Proof. Consider a CI statement in $\mathcal{L}^C$ of the form

$$X \perp\!\!\!\perp \mathbf{S}^+ \setminus Pa(\mathbf{C}) \mid Pa(\mathbf{C}) \setminus \{X\}, \text{ where}$$

$$\mathbf{S}^+ = \mathbf{V}^{\preceq X}_{\prec} \setminus De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C}))$$

for some variable $X \in \mathbf{V}^\prec$ and AC $\mathbf{C} \in \mathcal{AC}_X$. By Def. 4, there exists an ancestral set $\mathbf{S} \in \mathcal{S}_X$ such that $\mathbf{C} = \mathcal{C}(X)_{\mathcal{G}_\mathbf{S}}$. By Props. B.2.3 and B.2.1, $\mathbf{S}^+$ is ancestral and $\mathbf{C} = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{S}^+}}$.

First, we construct an ordering $\mathbf{V}^{\prec_*}$ under which $\mathbf{C} = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{V}^{\preceq X}_{\prec_*}}}$ using a 'pivot' technique about X. Given $\prec^*, \mathbf{S}^+$, initialise $\prec_* = \prec$. We re-order $\prec_*$ as follows. Let the pivot $P = X$. For each $Y \in \mathbf{V}^{\preceq X}_{\prec} \setminus \mathbf{S}^+$ in order of $\prec$, move Y to immediately succeed P in $\prec$ and update $P = Y$. Then, $\prec_*$ is a valid ordering since $\mathbf{S}^+$ is ancestral. Moreover, $\mathbf{S}^+ = \mathbf{V}^{\preceq X}_{\prec_*}$ and hence $\mathbf{C} = \mathcal{C}(X)_{\mathcal{G}_{\mathbf{V}^{\preceq X}_{\prec_*}}}$.

By definition, $\mathcal{L}^P$ contains the CI

$$X \perp\!\!\!\perp \mathbf{V}^{\preceq X}_{\prec_*} \setminus (Pa^+(\mathbf{C}) \cup \{X\}) \mid Pa^+(\{V_i\})$$

where $Pa^+(\{V_i\}) = Pa\Big(\mathcal{C}(V_i)_{\mathcal{G}_{\mathbf{V}^{\preceq V_i}_{\prec_*}}}\Big)_{\mathcal{G}_{\mathbf{V}^{\preceq V_i}_{\prec_*}}}$. Thus, the given CI from $\mathcal{L}^C$ has an identical counterpart in $\mathcal{L}^P$.

The graph $\mathcal{G}$ in Fig. D.2.1a with the ordering $X_1 \prec X_2 \prec X_4 \prec X_3$ provides an example where $\mathcal{L}^C \subsetneq \mathcal{L}^P$. We have $\mathcal{L}^C = \{X_4 \perp\!\!\!\perp \{X_1, X_2\}\}$. However, $\mathcal{L}^P = \{X_4 \perp\!\!\!\perp \{X_1, X_2\}, X_4 \perp\!\!\!\perp \{X_2\}, X_1 \perp\!\!\!\perp \{X_4\} \mid \{X_2\}, X_1 \perp\!\!\!\perp \{X_4\}, X_2 \perp\!\!\!\perp \{X_4\} \mid \{X_1\}\}$, as shown in Ex. D.2.2. □

We reproduce Fig. D.2.2 to demonstrate the 'pivot' technique used in the proof above.

■ **Example D.2.3.** Continuing Ex. 7, we demonstrate the construction used in the proof of Prop.D.2.1 for the graph $\mathcal{G}$ in Fig. D.2.2. We fix the ordering $A \prec B \prec C \prec D \prec E \prec F \prec H \prec I \prec X \prec J \prec K$ for C-LMP.

1. Fig. D.2.2a depicts the CI $X \perp\!\!\!\perp \{C, D, E, F\} \mid \{B\}$. Here, $\mathbf{C} = Pa(\mathbf{C}) = \{X, B\}, \mathbf{S}^+ = \{B, C, D, E, F, X\}$. We construct the ordering $\prec_*$: $B \prec C \prec D \prec E \prec F \prec X \prec A \prec H \prec I \prec J \prec K$.
2. Fig. D.2.2b depicts the CI $X \perp\!\!\!\perp \{A, D, I\} \mid \{H\}$. Here, $\mathbf{C} = Pa(\mathbf{C}) = \{X, H\}, \mathbf{S}^+ = \{A, D, H, I, X\}$. We construct the ordering $\prec_*$: $A \prec D \prec H \prec I \prec X \prec B \prec C \prec E \prec F \prec J \prec K$.
3. Fig. D.2.2c depicts the CI $X \perp\!\!\!\perp \{A, F, I\} \mid \{H, E\}$. Here, $\mathbf{C} = Pa(\mathbf{C}) = \{X, H, E\}, \mathbf{S}^+ = \{A, E, F, H, I, X\}$. We construct the ordering $\prec_*$: $A \prec E \prec F \prec H \prec I \prec X \prec B \prec C \prec D \prec J \prec K$.

Each ordering $\prec_*$ constructed for the given $\mathbf{C}, \mathbf{S}^+$ implies $\mathbf{V}^{\preceq X}_{\prec_*} = \mathbf{S}^+$ and $Pa^+(\{X\}) = Pa(\mathbf{C}) \setminus \{X\}$ in Def. D.2.3. □

Prop. D.2.1 thus implies the following corollary.

Corollary D.2.1. *Let $\mathcal{G}$ be a causal graph, $\mathbf{V}^\prec$ a consistent ordering, and $P(\mathbf{v})$ a probability distribution over the set of variables $\mathbf{V}$. Then, the following conditions are equivalent.*

(G) *$P(\mathbf{v})$ satisfies the global Markov property for $\mathcal{G}$.*
(L) *$P(\mathbf{v})$ satisfies the c-component local Markov property for $\mathcal{G}$ with respect to $\mathbf{V}^\prec$.*
(F) *$P(\mathbf{v})$ is semi-Markov relative to $\mathcal{G}$.*

Proof. The equivalence of (G) and (L) follows from Thm. 1.

(G) $\implies$ (F). Given a DAG $\mathcal{G}$ and a distribution $P(\mathbf{v})$, we need to show $P(\mathbf{v})$ factorizes according to Def. D.2.2 for every ordering $\mathbf{V}^\prec$ consistent with $\mathcal{G}$. Fix an arbitrary ordering $\mathbf{V}^\prec$. Using the chain rule, we factorize

$$P(\mathbf{v}) = \prod_{V_i \in \mathbf{V}^\prec} p(v_i \mid v_1, \ldots, v_{i-1})$$

Then, let $Pa^+(\{V_i\}) = Pa\Big(\mathcal{C}(V_i)_{\mathcal{G}_{\mathbf{V}^{\preceq V_i}}}\Big)_{\mathcal{G}_{\mathbf{V}^{\preceq V_i}}} \setminus \{V_i\}$. It suffices to show that

$$V_i \perp_d \mathbf{V}^{\preceq V_i} \setminus (Pa^+(\{V_i\}) \cup \{V_i\}) \mid Pa^+(\{V_i\}) \text{ in } \mathcal{G}$$

Since $\mathbf{V}^{\preceq V_i}$ is an ancestral set, $\mathbf{C} = \mathcal{C}(V_i)_{\mathcal{G}_{\mathbf{V}^{\preceq V_i}}}$ is an AC relative to V_i. By definition, $Sp(\mathbf{C}) \setminus \mathbf{C} = \emptyset$, hence $\mathbf{V}^{\preceq V_i} \setminus De(Sp(\mathbf{C}) \setminus Pa(\mathbf{C})) = \mathbf{V}^{\preceq V_i}$. By Prop. C.1.1, we get the required d-separation. Since $P(\mathbf{v})$ satisfies the global Markov property for $\mathcal{G}$, this d-separation implies that

$$V_i \perp\!\!\!\perp \mathbf{V}^{\preceq V_i} \setminus (Pa^+(\{V_i\}) \cup \{V_i\}) \mid Pa^+(\{V_i\}) \text{ in } P(\mathbf{v}).$$

This allows us to simplify the factorisation of $P(\mathbf{v})$ to

$$P(\mathbf{v}) = \prod_{V_i \in \mathbf{V}^{\prec}} p(v_i \mid pa_i^+)$$

(F) $\implies$ (C). If $P(\mathbf{v})$ is semi-Markov relative to $\mathcal{G}$, then each of the semi-Markov relative CIs of $\mathcal{G}$ (Def. D.2.3) must hold in $P(\mathbf{v})$. Since the C-LMP CIs of $\mathcal{G}$with respect to $\mathbf{V}^{\prec}$ are a subset of the semi-Markov relative CIs (Prop. D.2.1), the C-LMP CIs must necessarily hold in $P(\mathbf{v})$. □

D.3 No Reduction to the Markovian Case

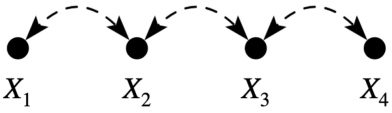
In this section, we consider another possible approach to testing semi-Markovian graphs, which attempts to reduce the problem to the Markovian setting.

To elaborate, say we want to test if a given semi-Markovian graph $\mathcal{G}$ is consistent with a given observational distribution $P(\mathbf{v})$. Consider an approach which maps $\mathcal{G}$ to an 'equivalent' Markovian graph $\mathcal{G}_M$, and tests $\mathcal{G}_M$. The sense of 'equivalence' must be such that any given distribution $P(\mathbf{v})$ is consistent with $\mathcal{G}$ if and only if it is consistent with $\mathcal{G}_M$. In other words, $\mathcal{G}$ and $\mathcal{G}_M$ must encode exactly the same CIs over the observed variables. This hypothetical reduction would mean that to test $\mathcal{G}$, we can test $\mathcal{G}_M$, which admits a simpler local Markov property.

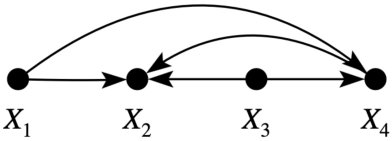
However, such a reduction is not possible in general, as the following example shows. Semi-Markovian graphs can represent strictly more conditional independence structures than Markovian graphs. Not every semi-Markovian graph can be mapped to an equivalent (in the above sense) Markovian graph.

■ **Example D.3.1** (Markovian graphs can not represent all conditional independencies among observables (Bareinboim et al. 2022, Ex. 16))**.** Consider the semi-Markovian graph $\mathcal{G}$ shown in Fig. D.3.1a, with consecutive colliders X_2 and X_3. $\mathcal{G}$ encodes the CIs $X_1 \perp\!\!\!\perp X_3$; $X_1 \perp\!\!\!\perp X_4$; $X_2 \perp\!\!\!\perp X_4$; $X_1 \perp\!\!\!\perp X_3 \mid X_4$; $X_1 \perp\!\!\!\perp X_4 \mid X_3$; $X_1 \perp\!\!\!\perp X_4 \mid X_2$, and no more. Any variables X_i and X_{i+1} can not be separated given any observed variable due to the presence of unobservd confounding between them. Thus, X_i and X_{i+1} must be adjacent in any Markovian graph $\mathcal{G}_M$ encoding the same CIs as $\mathcal{G}$. That is, $\mathcal{G}_M$ must have adjacencies $X_1 - X_2 - X_3 - X_4$. To encode $X_1 \perp X_3$, the only option is to orient $X_1 \to X_2 \leftarrow X_3$. To encode $X_2 \perp X_4$, the only option is to orient $X_2 \to X_3 \leftarrow X_4$. Therefore, $\mathcal{G}_M$ can not simultaneously encode $X_1 \perp X_3$ and $X_2 \perp X_4$, both of which are encoded in $\mathcal{G}$. This shows that any Markovian $\mathcal{G}_M$ can not encode exactly the same CIs as $\mathcal{G}$.

Say we orient $X_1 \to X_2 \leftarrow X_3$ in $\mathcal{G}_M$. If we additionally orient $X_3 \to X_4$, this implies $X_1 \perp\!\!\!\perp X_3$ and $X_1 \perp\!\!\!\perp X_4$ but not $X_2 \perp\!\!\!\perp X_4$. Moreover, it further implies $X_2 \perp\!\!\!\perp X_4 \mid X_3$, which does not hold in $\mathcal{G}$. To cancel this implication, we can add an edge $X_4 \to X_2$ to $\mathcal{G}_M$, which gives up on the CI $X_4 \perp\!\!\!\perp X_2$ encoded in $\mathcal{G}$. Still, $\mathcal{G}_M$ implies $X_1 \perp\!\!\!\perp X_4 \mid \{X_2, X_3\}$, which does not hold in $\mathcal{G}$. This requires an additional edge, say $X_1 \to X_4$. The resulting $\mathcal{G}_M$ (Fig. D.3.1b) encodes only one CI: $X_1 \perp\!\!\!\perp X_3$. In fact, any sequence of choices made in this construction can not result in a Markovian graph encoding exactly the same CIs as $\mathcal{G}$.



(a) A semi-Markovian graph $\mathcal{G}$ for which there exists no Markovian graph encoding exactly the same CIs.



(b) A Markovian graph $\mathcal{G}_M$ which encodes a subset of those CIs encoded in $\mathcal{G}$. Removing any edge from $\mathcal{G}_M$ would violate this condition.

Figure D.3.1: Graphs associated with Ex. D.3.1.

D.4 Examples

The following example shows that the total number of vacuous CIs invoked by C-LMP may be exponential with respect to the number of nodes in the input graph.

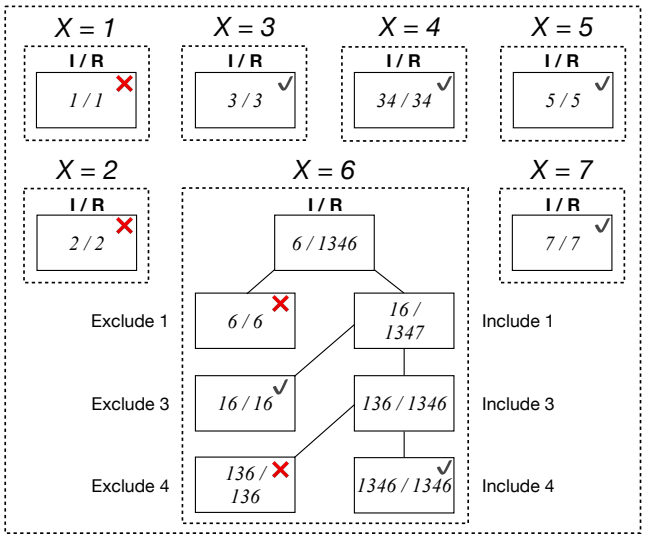
■ **Example D.4.1.** Consider the three causal graphs in Fig. D.4.1, each of which is a bidirected clique encoding no CIs. For $\mathcal{G}^{b1}$ shown in Fig. D.4.1a, C-LMP invokes 7 vacuous CIs: $A_1 \perp\!\!\!\perp \emptyset$, $A_2 \perp\!\!\!\perp \emptyset$, $A_2 \perp\!\!\!\perp \emptyset \mid \{A_1\}$, $A_3 \perp\!\!\!\perp \emptyset$, $A_3 \perp\!\!\!\perp \emptyset \mid \{A_1\}$, $A_3 \perp\!\!\!\perp \emptyset \mid \{A_2\}$, $A_3 \perp\!\!\!\perp \emptyset \mid \{A_1, A_2\}$. $\mathcal{G}^{b2}$ shown in Fig. D.4.1b is constructed by adding one variable A_4 to $\mathcal{G}^{b1}$ with three bidirected edges $A_4 \leftrightarrow A_i$ for $1 \leq i \leq 3$. C-LMP invokes 15 vacuous CIs for $\mathcal{G}^{b2}$. If we add another variable A_5 to $\mathcal{G}^{b2}$ with four bidirected edges $A_5 \leftrightarrow A_i i$ for $1 \leq i \leq 4$, C-LMP invokes 31 vacuous CIs. As shown in Fig. D.4.1c which generalizes this pattern to variables $\{A_1, \cdots, A_n\}$ with bidirected edges between every A_i and A_j with $1 \leq i, j \leq n, i \neq j$, C-LMP invokes $2^n - 1$ vacuous CIs. □

The following example expands on Ex. 10. We demonstrate the execution of LISTCI($\mathcal{G}^3, \mathbf{V}^{\prec}$) with $\mathcal{G}^3$ shown in Fig. 3b and $\mathbf{V}^{\prec} = \{A, B, C, D, E, F, H, J\}$. The full search tree for the execution of LISTCI (in Ex. 9) is given in Fig. D.2.3.

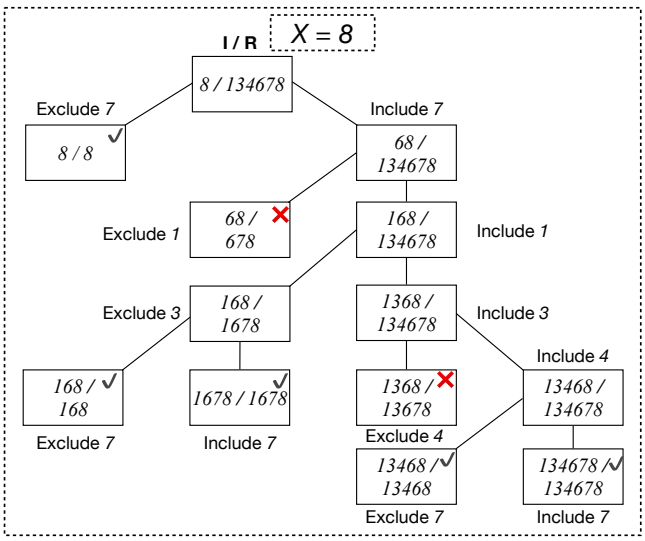
■ **Example D.4.2.** Expanding Ex. 10. Let $\mathcal{G}^3$ be the causal graph shown in Fig. 3b and $\mathbf{V}^{\prec} = \{A, B, C, D, E, F, H, J\}$. We show a part of running LISTCI($\mathcal{G}^3, \mathbf{V}^{\prec}$) with $X = J$ starting from the root node $\mathcal{N}(\{J\}, \{A, C, D, F, H, J\})$ to the leaf node $\mathcal{L}(\{A, F, J\}, \{A, F, J\})$.

Initially, the search starts from $\mathcal{N}$ which is constructed at line 5 of LISTCI with $X = J$, $\mathbf{I} = \{J\}$ and $\mathbf{R} = \{A, C, D, F, H, J\}$. At line 3 of LISTCIX, FINDAAC returns $\{J\}$. With $s = F$ and $\mathbf{R}' = \{J\}$, the recursive call LISTCIX($\mathcal{G}^3, J, \mathbf{V}^{\prec}, \{J\}, \{J\}$) is made at line 10, spawning a child $\mathcal{N}_1(\{J\}, \{J\})$. The search continues from $\mathcal{N}_1$. FINDAAC returns $\{J\}$. $\mathcal{N}_1$ is a leaf node, and LISTCIX outputs a CI: $J \perp\!\!\!\perp \{A, B, C, D, E\}$ at line 6.

After, LISTCIX backtracks to the parent $\mathcal{N}$. Then, with $\mathbf{I}' = \{F, J\}$ constructed at line 9, a recursive call LISTCIX($\mathcal{G}^3, J, \mathbf{V}^{\prec}, \{F, J\}, \{A, C, D, F, H, J\}$) is made at line 10, spawning a child $\mathcal{N}_2(\{F, J\}, \{A, C, D, F, H, J\})$.

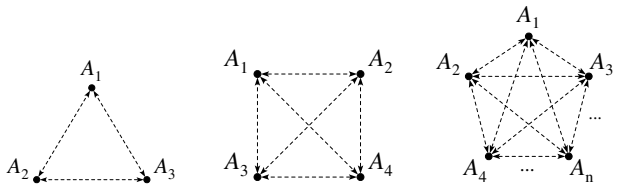


(a) A set of search trees, one for each $X \neq J$



(b) A search tree for $X = J$

Figure D.2.3: A set of search trees illustrating the running of LISTCI in Ex. 9.



(a) $\mathcal{G}^{b1}$ (b) $\mathcal{G}^{b2}$ (c) $\mathcal{G}^{b3}$

Figure D.4.1: Three examples to demonstrate that total number of vacuous CIs invoked by C-LMP may be exponential with respect to the number of nodes in a graph.

At $\mathcal{N}_2$, FINDAAC returns $\{A, F, J\}$. With $s = A$ and $\mathbf{R}' = \{F, H, J\}$, another recursive call LISTCIX($\mathcal{G}^3, J, \mathbf{V}^{\prec}, \{F, J\}, \{F, H, J\}$) is made at line 10, spawning a child $\mathcal{N}_3(\{F, J\}, \{F, H, J\})$. At $\mathcal{N}_3$, FINDAAC returns $\perp$, backtracking to $\mathcal{N}_2$. with $\mathbf{I}' = \{A, F, J\}$, a recursive call LISTCIX($\mathcal{G}^3, J, \mathbf{V}^{\prec}, \{A, F, J\}, \{A, C, D, F, H, J\}$) creates a child $\mathcal{N}_4(\{A, F, J\}, \{A, C, D, F, H, J\})$. The recursion continues in the following order: $\mathcal{N}_4$ adds a child $\mathcal{N}_5(\{A, F, J\}, \{A, F, H, J\})$ with $s = C$ and $\mathbf{R}' = \{A, F, H, J\}$, and $\mathcal{N}_5$ adds a child $\mathcal{N}_6(\{A, F, J\}, \{A, F, J\})$ with $s = H$ and $\mathbf{R}' = \{A, F, J\}$. $\mathcal{N}_6 = \mathcal{L}$ is a leaf node and FINDAAC returns $\{A, F, J\}$. Finally, LISTCIX outputs a CI: $J \perp\!\!\!\perp \{B\} \mid \{A, F\}$ at line 6. □

E Further Results

We present a procedure LISTGMP (Fig. E.0.1) that lists all CIs invoked by GMP for a causal graph $\mathcal{G}$ over a set of variables $\mathbf{V}$. The following result states that LISTGMP correctly lists all such CIs.

1: **function** LISTGMP($\mathcal{G}, \mathbf{V}$)
2: **Output:** Listing CIs invoked by GMP for $\mathcal{G}$ over $\mathbf{V}$.
3: **for** each $\mathbf{X}$ with $\emptyset \subset \mathbf{X} \subset \mathbf{V}$ **do**
4: **for** each $\mathbf{Y}$ with $\emptyset \subset \mathbf{Y} \subseteq \mathbf{V} \setminus \mathbf{X}$ **do**
5: **for** $\mathbf{Z}$ with $\emptyset \subseteq \mathbf{Z} \subseteq \mathbf{V} \setminus (\mathbf{X} \cup \mathbf{Y})$ **do**
6: Output $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$
7: **end function**

Figure E.0.1: A function that lists all CIs invoked by GMP.

Lemma E.0.1 (Correctness of LISTGMP)**.** *Given a causal graph $\mathcal{G}$ over a set of variables $\mathbf{V}$,* LISTGMP($\mathcal{G}, \mathbf{V}$) *lists all and only all conditional independence relations invoked by the global Markov property for $\mathcal{G}$.*

Proof. The proof follows by construction from Def. 2. □

F Experimental Details

All experiments were run on a machine with CPU: Apple M2 Chip, 16GB of RAM, and macOS operating system. We used a single core for the experiments. The algorithms are implemented in Python.

This section is organized as follows. Section F.1 presents details of the runtime of LISTCI and other algorithms. Section F.2 shows detailed result on testing a hypothesized model against a real-life dataset. Section F.3 provides detailed analysis of the total number of non-vacuous CIs invoked by C-LMP. We use LISTCI for the analysis.

F.1 Comparison of LISTCI with Other Algorithms

We compare the runtime of LISTCI with other two algorithms: LISTGMP and LISTCIBF over `bnlearn` instances.

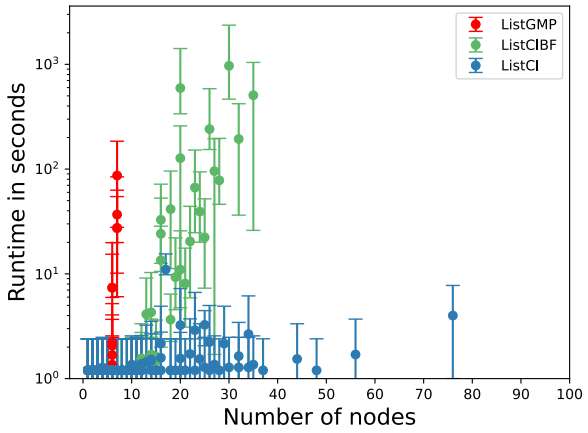


Figure F.1.1: Plot of runtimes of the algorithms LISTGMP, LISTCIBF, and LISTCI on graphs of various sizes. Each dot represents the average runtime of over 10 sample graphs for a given number of nodes and projection level $U \in \{0, 10, 20, \ldots, 90\}$. Lower and upper limits of an error bar represent the 1st and 3rd quartiles respectively. The y-axis uses a logarithmic scale.

The runtime of the algorithms across different levels of projection $U \in \{0, 20, 40, 60, 80\}$ respectively, are shown in Tables F.1.1 - F.1.5.

Fig. F.1.1 depicts further statistics on the results shown in Fig. 5. Each point corresponds to average runtime of the specified algorithm on a given graph and projection level $U \in \{0, 10, 20, \ldots, 90\}$, from which 10 random projections were generated. Only the cases where all 10 samples did not time out ($>$ 1 hour) are shown. Each dot is associated with an error bar where the 1st and 3rd quartiles across the 10 samples are given by the the lower and upper limits, respectively.

F.2 Application to Model Testing

In this section, we provide more details on our application of LISTCI to the task of model testing in Section 5. Recall that we test an expert-provided ground-truth DAG (11 nodes and 16 edges, shown in Fig. F.2.1) against a real-world protein signaling dataset with 853 samples (Sachs et al. 2005). We present the details in the following example.

■ **Example F.2.1.** Let $\mathcal{G}$ be the ground-truth DAG shown in Fig. F.2.1, and fix the consistent ordering $\mathbf{V}^{\prec} = \{PKA, PIP3, Plcg, Akt, PIP2, PKC, Raf, P38, Jnk, Mek, Erk\}$. GMP invokes 76580 CIs for $\mathcal{G}$. A naive approach would be to test all these CIs against the data. In contrast, C-LMP invokes 10 CIs for $\mathcal{G}$ with respect to $\mathbf{V}^{\prec}$, which together imply all CIs of the GMP. Therefore, C-LMP makes it possible to test the CIs encoded in $\mathcal{G}$ against the data. The full list of CIs that C-LMP invokes is shown in Table F.2.1.

To generate and test these CIs, we call LISTCI($\mathcal{G}, \mathbf{V}^{\prec}$) and use a kernel-based CI test from the `causal-learn` package (Zheng et al. 2024) with p-value $p = 0.05$ (for the null hypothesis of independence).

As shown in Table F.2.1, seven out of ten CIs invoked by C-LMP resulted in $p > 0.05$. □

Graphs			Runtime (mm:ss)		
Name	n	m	LISTGMP	LISTCIBF	LISTCI
asia	8	8	03:49	00:00	00:00
cancer	5	4	00:00	00:00	00:00
earthquake	5	4	00:00	00:00	00:00
sachs	11	17	-	00:00	00:00
survey	6	6	00:01	00:00	00:00
alarm	37	46	-	-	00:01
barley	48	84	-	-	00:01
child	20	25	-	00:46	00:00
insurance	27	52	-	00:54	00:00
mildew	35	46	-	04:10	00:00
water	32	66	-	-	00:00
hailfinder	56	66	-	-	00:01
win95pts	76	112	-	-	00:02

Table F.1.1: No variables unobserved.

Graphs			Runtime (mm:ss)		
Name	n	m	LISTGMP	LISTCIBF	LISTCI
asia	7	7	00:14	00:00	00:00
cancer	4	3	00:00	00:00	00:00
earthquake	4	3	00:00	00:00	00:00
sachs	9	14	-	00:00	00:00
survey	5	5	00:00	00:00	00:00
alarm	30	40	-	-	00:00
barley	39	88	-	-	00:01
child	16	24	-	00:05	00:00
insurance	22	57	-	00:06	00:00
mildew	28	45	-	00:29	00:00
water	26	78	-	13:22	00:01
hailfinder	45	84	-	-	06:06
win95pts	61	111	-	-	00:35

Table F.1.2: 20% of variables unobserved.

Graphs			Runtime (mm:ss)		
Name	n	m	LISTGMP	LISTCIBF	LISTCI
asia	5	6	00:00	00:00	00:00
cancer	3	2	00:00	00:00	00:00
earthquake	3	2	00:00	00:00	00:00
sachs	7	12	00:02	00:00	00:00
survey	4	4	00:00	00:00	00:00
alarm	23	35	-	-	00:00
barley	29	91	-	02:29	00:01
child	12	25	-	00:01	00:00
insurance	17	65	-	00:06	00:02
mildew	21	39	-	00:04	00:00
water	20	78	-	00:21	00:01
hailfinder	34	67	-	-	00:12
win95pts	46	98	-	-	02:07

Table F.1.3: 40% of variables unobserved.

Graphs			Runtime (mm:ss)		
Name	n	m	LISTGMP	LISTCIBF	LISTCI
asia	4	3	00:00	00:00	00:00
cancer	2	1	00:00	00:00	00:00
earthquake	2	1	00:00	00:00	00:00
sachs	5	7	00:00	00:00	00:00
survey	3	2	00:00	00:00	00:00
alarm	15	27	-	00:10	00:00
barley	20	80	-	04:18	00:01
child	8	15	-	00:00	00:00
insurance	11	47	-	00:00	00:00
mildew	23	20	-	00:03	00:00
water	13	47	-	00:01	00:00
hailfinder	23	42	-	-	00:01
win95pts	31	53	-	-	00:14

Table F.1.4: 60% of variables unobserved.

Graphs			Runtime (mm:ss)		
Name	n	m	LISTGMP	LISTCIBF	LISTCI
asia	2	1	00:00	00:00	00:00
cancer	1	0	00:00	00:00	00:00
earthquake	1	0	00:00	00:00	00:00
sachs	3	2	00:00	00:00	00:00
survey	2	1	00:00	00:00	00:00
alarm	8	10	-	00:00	00:00
barley	10	23	-	00:01	00:01
child	4	6	00:01	00:00	00:00
insurance	6	20	00:02	00:00	00:00
mildew	7	14	00:19	00:00	00:00
water	7	12	00:49	00:00	00:00
hailfinder	12	24	-	00:05	00:00
win95pts	16	18	-	02:14	00:01

Table F.1.5: 80% of variables unobserved.

Table F.1.1 - F.1.5: Summary of runtime of algorithms over various graphs. For each graph, the stated percent of variables were randomly chosen as unobserved, and the graph was projected onto the remaining observed variables. Runtime is rounded to the nearest integer (second). The symbol "-" indicates that the algorithm took over an hour on at least one sample graph.

The test results show that $\mathcal{G}$ may need to be revised, and the exact list of CIs that are violated may help experts in the revision process. However, we note that significance testing (whether rejecting the null hypothesis or not) has its own limitations. For example, selection of the level of significance impacts the probability of Type I error, and sample size of the dataset affects the likelihood of Type II error, especially for small datasets.

F.3 Analysis of C-LMP

In this section, we use LISTCI to understand the total number of non-vacuous CIs invoked by C-LMP. Let **CI** denote this number. We showed that **CI** is bounded by $\Theta(n2^s)$ for a DAG with n nodes whose largest c-component has size s. While we give a concrete DAG to show this bound is tight, our hope is to empirically evaluate how often this worst-case arises, and **CI** in the 'average case' using random graphs. LISTCI makes such empirical analysis of C-LMP possible by giving a way to efficiently compute **CI**.

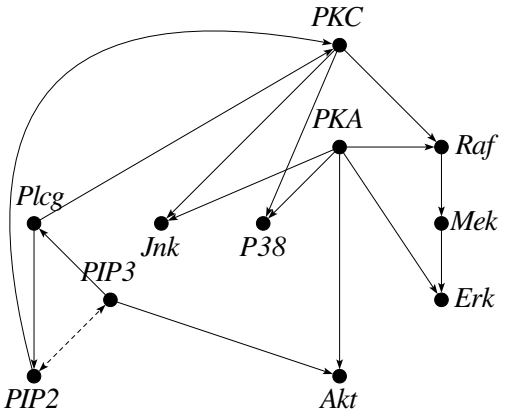


Figure F.2.1: The ground-truth DAG (a protein-signaling network) sn (Sachs et al. 2005, Fig. 2).

Hypothesis. We hypothesize that the following two parameters are key in determining **CI** for a DAG $\mathcal{G}$:

1. mu: the number of bidirected edges in $\mathcal{G}$, and
2. s: the size of the largest c-component in $\mathcal{G}$.

Intuitively, both the size of c-components in $\mathcal{G}$ and their sparsity, which depends on mu, are important indicators of **CI**. Both parameters control the number of subsets of the c-component that result in admissible ancestral c-components.

Random graphs. We run LISTCI on random graphs to understand **CI** in the average case. In particular, we use a minor variant of Erdős-Rényi random graphs to include both directed and bidirected edges. We define a random causal DAG as $\mathcal{G}(n, pd, pb)$ where pd (pb) represents the probability of a directed (bidirected) edge between a given pair of nodes. Each possible edge is an independent Bernoulli.

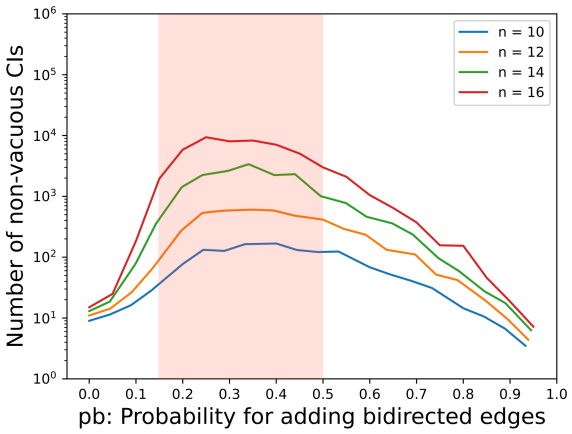
Experimental design. For each experiment, we fix our controls: n, the number of nodes, and md, the number of directed edges, in $\mathcal{G}$. We test $md = 0, n, 2n$ for select $n \in [10, 50]$. Then, we change mu, and observe s and **CI**. For each n and md, we ran each experiment on 100 sample graphs. Each run of LISTCI was given an hour until timeout. Only complete runs are shown.

Results and discussion.

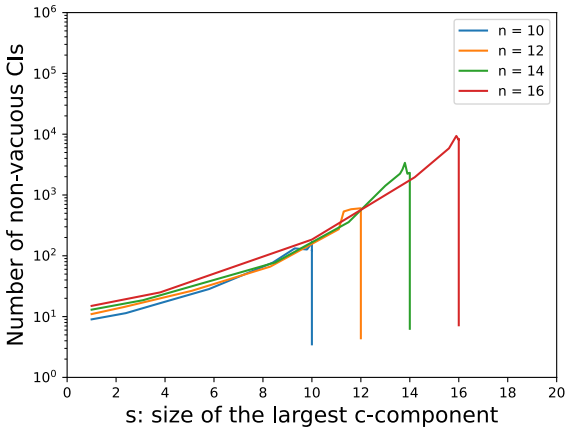
1. **Case 1**: $md = 0$. First, for simplicity, we work with small graphs containing no directed edges. Starting from $pb = 0$ (or $mu = 0$), we incrementally add bidirected edges until reaching full capacity, i.e., $pb = 1$ or $mu = \frac{n(n-1)}{2}$. Then, an interesting trend emerges as shown in Fig. F.3.1a. Roughly speaking, there are two phases seen on the curve.

CIs implied by $\mathcal{G}$	p-value
$PIP3 \perp\!\!\!\perp PKA$	0.175
$Plcg \perp\!\!\!\perp PKA \mid PIP3$	0.081
$Akt \perp\!\!\!\perp Plcg \mid PIP3, PKA$	0.370
$PIP2 \perp\!\!\!\perp Akt, PKA \mid PIP3, Plcg$	0.648
$PKC \perp\!\!\!\perp Akt, PIP3, PKA \mid PIP2, Plcg$	0.318
$Raf \perp\!\!\!\perp Akt, PIP2, PIP3, Plcg \mid PKA, PKC$	0.036
$P38 \perp\!\!\!\perp Akt, PIP2, PIP3, Plcg, Raf \mid PKA, PKC$	0.680
$Jnk \perp\!\!\!\perp Akt, P38, PIP2, PIP3, Plcg, Raf \mid PKA, PKC$	0.002
$Mek \perp\!\!\!\perp Akt, Jnk, P38, PIP2, PIP3, PKA, PKC, Plcg \mid Raf$	0.544
$Erk \perp\!\!\!\perp Akt, Jnk, P38, PIP2, PIP3, PKC, Plcg, Raf \mid Mek, PKA$	0.000

Table F.2.1: Summary of results on testing a ground-truth DAG against a protein-signaling dataset. A kernel-based CI test was used to test the set of CIs (invoked by C-LMP) on the dataset. P-value is rounded to the nearest three digits after the decimal point.



(a) pb and **CI**



(b) s and **CI**

Figure F.3.1: Illustration of results in Case 1 ($md = 0$). (a) displays two-phase transitions: as pb increases, **CI** grows rapidly up to a certain point (Phase 1) but shrinks after (Phase 2). (b) shows that s correlates with **CI** only within Phase 1. The red box indicates the 'critical region.'

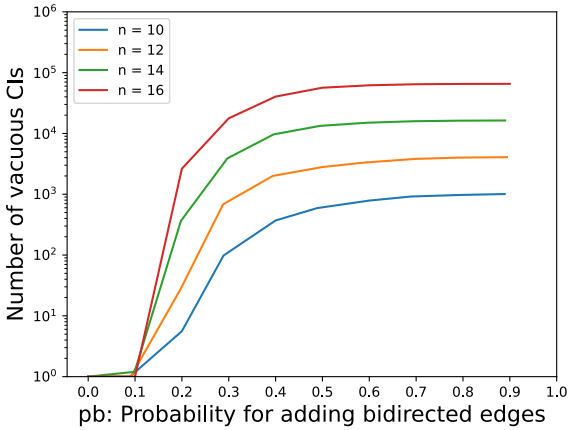
(a) Phase 1 on the left half of the curve. As more bidirected edges are added (i.e., pb increases), **CI** grows exponentially up to a certain peak region.

(b) Phase 2 on the right half of the curve. After reaching this peak, **CI** decreases exponentially as more bidirected edges are added.

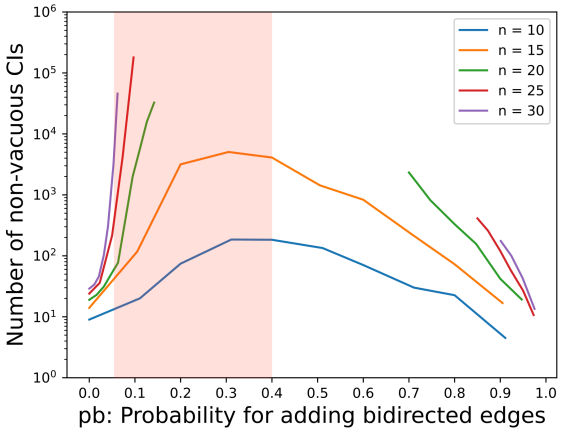
A possible explanation for the pattern shown in Phase 1 is that larger c-components tend to be constructed as pb increases. Then, s increases in general. As given by the bound $O(n2^s)$ (Prop 1), **CI** increases exponentially with a linear increase in s. The curve in Fig F.3.1b showing this relationship corresponds to Phase 1. Intuitively, a linear increase of the size of the largest c-component $\mathbf{C}$ (of size s) implies an exponential increase of total combination of subsets of $\mathbf{C}$ (i.e., MBs). As shown by Lemma B.3.1 and Thm. 2, each MB maps uniquely to each CI invoked by C-LMP. Thus, the total number of MBs is the sum of the numbers of vacuous and non-vacuous CIs, which is represented by the "sum" of both curves: one curve in Fig. F.3.1a and the other curve in Fig. F.3.2a, respectively.

On the other hand, in Phase 2, when even more bidirected edges are added to $\mathcal{G}$, large c-components (or in fact, the largest and only c-component of size n) may become more dense in terms of bidirected edges. In the extreme case with $pb = 1$, $\mathcal{G}$ becomes a bidirected clique of size n. With more bidirected paths between nodes, the number of d-separations in the graph decreases, leading to a decrease in **CI**. Conceptually, the total number of MBs increases as c-components get more dense. However, the ratio of MBs that result in non-vacuous CIs to total MBs decreases at a higher rate than the rate of increase of the number of MBs. We observe the difference by comparing Fig. F.3.1a and Fig. F.3.2a. This results in the decrease in **CI**.

In Fig F.3.1b, a vertical line where s stays constant, i.e., $s = n$, corresponds to Phase 2. We note that s being a constant is a natural consequence of the experimental setup. When pb continues to increase from 0, all nodes in $\mathcal{G}$ will eventually become connected to one another, and thus s converges to n. Once the point with $s = n$ is reached, s stays constant even with further addition of bidirected edges since the entire set of nodes in $\mathcal{G}$ is the largest and the only c-component in $\mathcal{G}$. When pb is further increased, the largest c-component in $\mathcal{G}$ becomes more dense, which explains the decrease in **CI**. Therefore, s may be a good indicator of **CI** in Phase 1, but not necessarily in Phase 2.

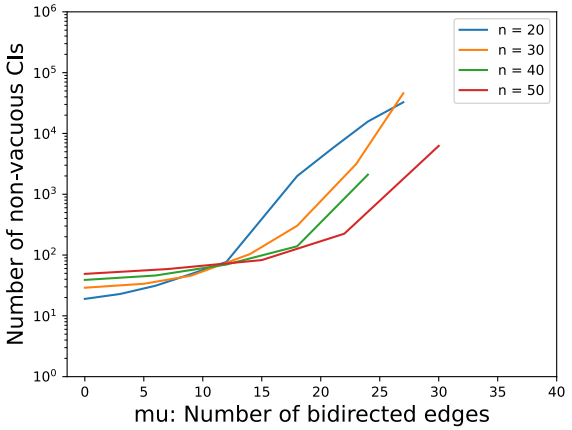


(a) pb and number of vacuous CIs

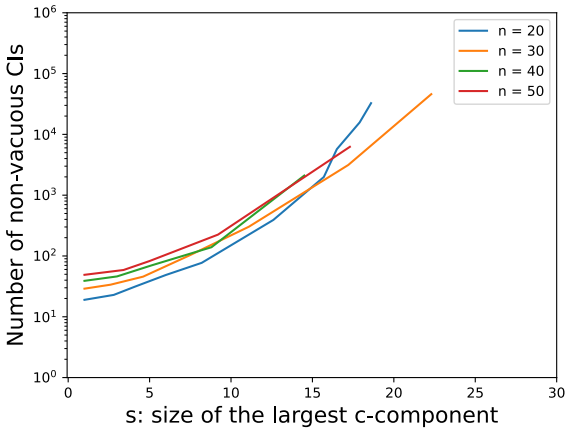


(b) pb and **CI**

Figure F.3.2: Illustration of results in Case 1. (a) Total number of vacuous CIs increases as $\mathcal{G}$ becomes more dense with respect to bidirected edges. (b) Two-phase transitions are not fully observable for $n \geq 20$. he red box indicates the 'critical region.'



(a) mu and **CI**



(b) s and **CI**

Figure F.3.3: Illustration of results in Case 1 within Phase 1. ListCI starts timing out at approximately $mu = 30$ or greater.

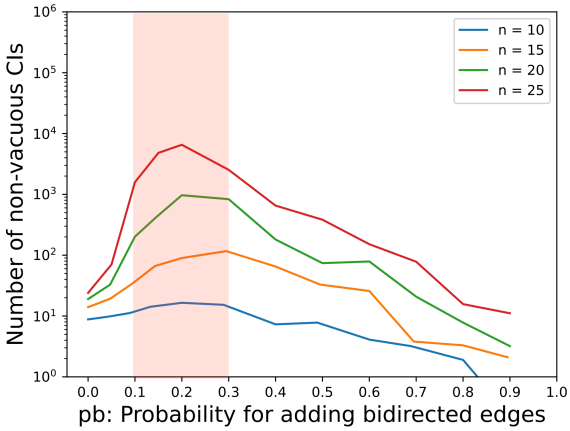
Another subtlety to note is the rate of growth of **CI** with respect to n. Within a "critical region" shown in Fig F.3.1a, observe gaps between the curves for each n. Even as n increases by two from $n = 10$, **CI** in this middle region grows exponentially. This may not be immediate as the bound on **CI** is linear in n. However, in the peak region of the curve, we have $s \approx n$, thus making **CI** exponential in n. This makes it infeasible to observe phase transitions over larger n in graphs without directed edges.

We verify the claim that phase transitions may not be fully observable for larger n. Bidirected edges are added slowly until ListCI starts timing out. The results are shown in Fig. F.3.3a. We see that $mu = 30$ is an approximate threshold after which ListCI may spend more than an hour. Given $mu = 30$, the threshold values of pb that correspond to each $n \in \{20, 30, 40, 50\}$ are $0.158, 0.069, 0.038$, and 0.024 respectively. Based on the curves shown in Fig. F.3.1a, it is possible ListCI times out before the peak. All curves for large n live within Phase 1.

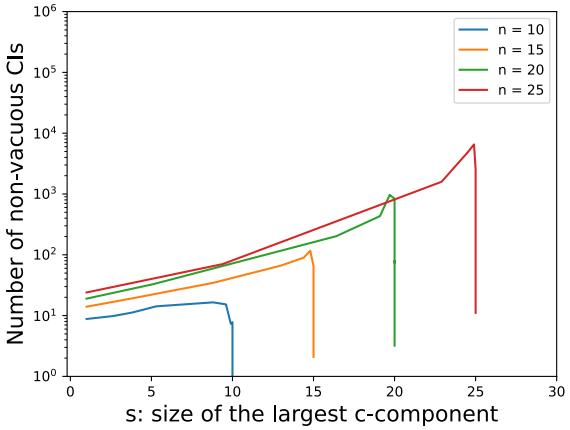
Additionally, we present Phase 2 for $n \in \{10, 15, 20, 25, 30\}$ in Fig. F.3.2b. Starting from $pb = 1$, we keep removing bidirected edges (i.e., decreasing pb) until ListCI starts timing out. For $n = 20$, ListCI times out with $pb < 0.7$. For $n = 30$, $pb < 0.85$ and for $n = 40$, $pb < 0.9$. All fraction of the curves represent some fraction of Phase 2.

Returning to s, we show the relationship between s and **CI** in Fig F.3.3b. As in the case for small n, **CI** is exponential in s during Phase 1.

It may seem that ListCI is not feasible on larger graphs. However, Case 1 considers an edge case with no directed

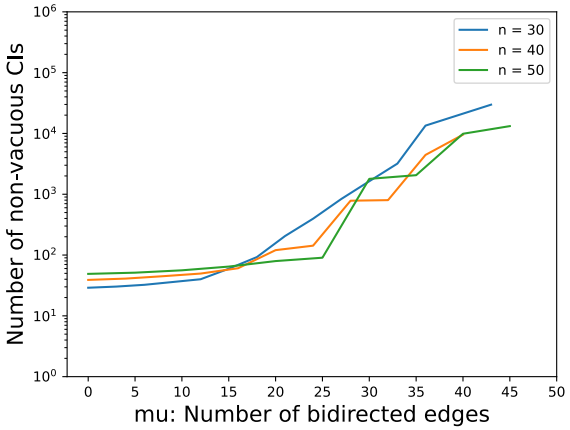


(a) pb and **CI**

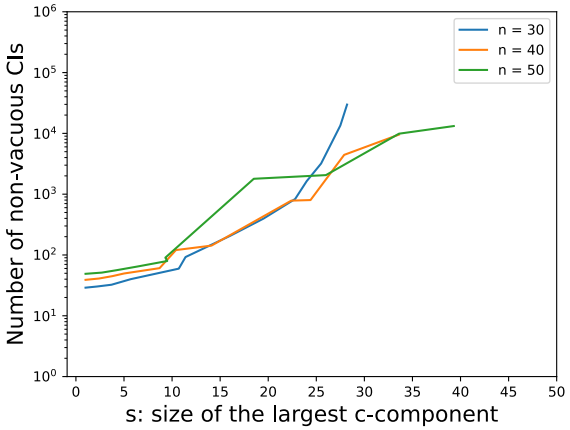


(b) s and **CI**

Figure F.3.4: Illustration of results in Case 2 ($md = n$). Overall, similar patterns are shown as in Case 1 (Fig. F.3.1). However, the rate of growth of **CI** with respect to n is lower than in Case 1. he red box indicates the 'critical region.'



(a) mu and **CI**



(b) s and **CI**

Figure F.3.5: Illustration of results in Case 2 by adding bidirected edges to a graph $\mathcal{G}$ across varying n. LISTCI starts timing out at approximately $mu = 50$ or greater.

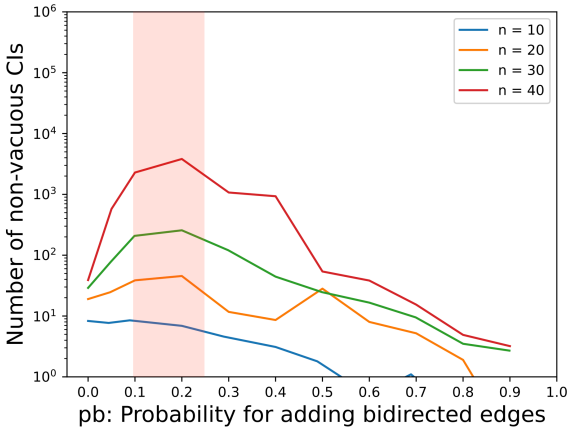
edges where all subsets of nodes are ancestral. The problem is highly unconstrained. Most real-world graphs are not this sparse, which makes **CI** less sensitive to changes in mu, as we explain in the next part.

2. **Case 2**: $md = n$. We use a similar setup as in Case 1, except that we add n directed edges on a wider range of graph sizes. When we incrementally add bidirected edges to $\mathcal{G}$ from $pb = 0$ up to $pb = 1$, a pattern identical to Case 1 (Fig. F.3.1a) arises in Case 2 (Fig. F.3.4a). A notable difference, however, is the rate of growth of **CI** with respect to n. For example, let $n = 20$. In Case 1 (Fig. F.3.3a), with increasing $mu > 20$, **CI** increases to 10^4 and beyond until LISTCI times out. On the other hand, in Case 2 (Fig. F.3.4a), **CI** does not exceed 10^3 for any mu (or pb). Still, the two-phase transition is not observable for larger n, i.e., $n \in \{30, 40, 50\}$. Similarly, as in Fig. F.3.1b, we observe an exponential relationship between s and **CI** (Fig. F.3.4b) in for $s < n$.

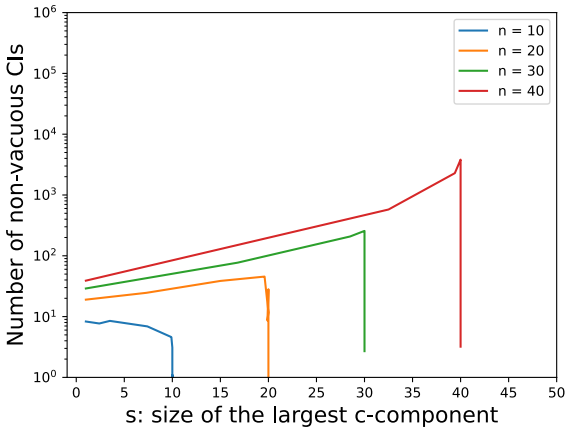
   Next, we let $n \in \{30, 40, 50\}$. The results are shown in Fig. F.3.5. We have a similar conclusion as in Case 1, except that an approximate threshold for mu until LISTCI times out is increased to 50. Given $mu = 50$, the values of pb that map to each $n \in \{30, 40, 50\}$ are $0.115, 0.064$, and 0.041 respectively. The larger threshold can be explained by the correspondence between ancestral sets and MBs. Since adding directed edges exponentially reduces the number of ancestral sets, this can only reduce the number of MBs, and hence **CI**. Inspecting the curves shown in Fig. F.3.4a, it is likely that LISTCI starts timing out before **CI** peaks.

3. **Case 3**: $md = 2n$.

   We continue the set up of Cases 1 and 2, now adding $2n$ directed edges to small-to-medium sized graphs. As



(a) pb and **CI**



(b) s and **CI**

Figure F.3.6: Illustration of results in Case 3 ($md = 2n$). Overall, the patters are similar as shown in Case 1 and Case 2 (Fig. F.3.1 and Fig. F.3.4). The rate of growth of **CI** with respect to n is lower than those in Case 1 and Case 2. he red box indicates the 'critical region.'

shown in Fig. F.3.6a, we see phase transitions for n up to 40. Comparing to Case 2 where $md = n$, the rate of growth of **CI** with mu is lower in general. For example, let $n = 20$. In Case 2 (Fig. F.3.4a), **CI** reaches approximately 10^3. However, in Case 3 (Fig. F.3.6a), **CI** does not reach 10^2, even in the peak. The relationship between s and **CI** seen in Cases 1 and 2 (Figs. F.3.1b and F.3.4b) – with two patterns corresponding to the two phases – is reproduced in Case 3 (Fig. F.3.6b).

Summarizing experimental findings from Cases 1, 2, and 3, we conclude that both the size s of the largest c-component $\mathbf{C}$ in $\mathcal{G}$ and the sparsity of $\mathbf{C}$ determined by the number of bidirected edges play a key role in **CI**. The reproducibility of the phase transitions and relationships between s, mu, and **CI** across different combinations of md and n lends credence to this conclusion.

G Frequently Asked Questions

Q1. Is it reasonable to expect that the causal graph is available? How do you get the graph?

Answer. The assumption of the causal diagram is made out of necessity; without causal assumptions, causal inferences are almost never possible (e.g., see the Causal Hierarchy Theorem in (Bareinboim et al. 2022, Section 1.3)).

In the real world, data scientists engage in causal modeling and leverage their background knowledge about the problem to construct a causal model (e.g., graph). Celebrated results in the literature, such as Pearl's do-calculus, were designed to take advantage of this knowledge in order to generate quantitative understanding of the system that was previously unknown to the data scientist. Part of the main theme in the field is about how to infer new facts given a collection of causal assumptions.

Against this context, the main goal of our work is to provide a set of tools to evaluate whether the assumptions encoded in a causal model are plausible, or formally compatible with the observed data. It is not easy to characterize or to list all of such assumptions, as discussed formally in Section 3 and empirically in Appendix F. We provide the first algorithm for listing a small set of CI assumptions in poly-delay, using which a model can be tested in settings with non-parametric distributions and arbitrary unobserved variables.

Finally, the task known as causal discovery aims to a coarser representation of the causal model from data, including from observational (Verma and Pearl 1992; Spirtes, Glymour, and Scheines 2001; Pearl 2000) and interventional data (Kocaoglu, Shanmugam, and Bareinboim 2017; Kocaoglu et al. 2019; Jaber et al. 2020; Li, Jaber, and Bareinboim 2023).

Q2. Can this result be used to evaluate the quality of a learned model, e.g., a partial ancestral graph (PAG) (Zhang 2008)?

Answer. Yes. If the learned model is a Markov equivalence class (MEC) of DAGs, for e.g., a PAG, all DAGs in the MEC imply exactly the same set of CIs. Therefore, an observational dataset is consistent with the MEC if and only if it is consistent with some (or every) DAG in the MEC. To test the learned MEC, one can choose any DAG in the MEC, and apply our result to this DAG.

Q3. There is already a simple local Markov property for Markovian models that is easy to enumerate. When you have a hidden variable model, can't you eliminate the hidden variables by mapping it to a Markovian model (i.e., Bayesian network) over the observed variables, and test CIs in the resulting Markovian model?

Answer. While appealing at first, such a reduction is not possible in general. For testing the Markovian model to be equivalent to testing the semi-Markovian model, they need to encode exactly the same CI constraints. However, semi-Markovian models can capture strictly more conditional independence structures than Markovian models. In other words, not every semi-Markovian model can be mapped to

an equivalent Markovian model. See Sec. D.3 for a simple example showing how this reduction is not possible.

Q4. What's the difference between (LMP,$\prec$) and C-LMP? Since they output an identical list of CIs, aren't they the same?

Answer. It is true that (LMP,$\prec$) and C-LMP invoke the same set of CIs. Since LISTCI lists CIs invoked by C-LMP, it thus equivalently lists CIs invoked by (LMP,$\prec$). There is nothing inherent in the definition of (LMP,$\prec$) (Def. A.1.3) that makes it impossible to list the CIs it invokes in poly-delay. However, in Def. A.1.2 and Def. A.1.3, MASs are defined non-constructively. Def. A.1.2 leaves it open whether there is exactly one MAS relative to an MB, and how to construct such an MAS. Therefore, each CI in (LMP,$\prec$) is also defined non-constructively. The only object with a constructive definition is the ancestral set, which is used to define MASs using universal quantifiers. This considerable degree of indeterminacy leads to the brute-force approach we develop in Section B.1. In contrast, the definition of C-LMP (Def. 5) is entirely constructive, and abstracts away the complexities of MASs and MBs. We give an explicit one-to-one mapping between ACs and CIs (Thm. 2) that does not need any universal quantifiers, except over the space of ACs. Therefore, the definition of C-LMP provides a natural path to enumerating the invoked CIs by enumerating ACs. Moreover, the explicit one-to-one mapping between ACs and CIs allows us to derive tight bounds on the number of CIs invoked by C-LMP (and equivalently, (LMP,$\prec$)) by reasoning about connected components in the graph, an approach that would not be clear from a non-constructive definition of CIs.

Q5. What happens if the total number of CIs invoked by C-LMP is exponential? Do we have to wait until LISTCI outputs the full list of CIs?

Answer. First, we note that C-LMP is an exponential improvement on the global Markov property (GMP) with respect to number of CIs invoked. In contrast with the $\Theta(4^n)$ many CIs invoked by GMP (Prop. C.3.1), C-LMP invokes $O(n2^s)$ number of CIs given a DAG on n variables whose largest c-component has size s (Prop. 1). The upshot is largest for a DAG with large n but small c-components. For instance, for the DAG $\mathcal{G}^2$ in Fig. 1b, GMP invokes 753 CIs but C-LMP invokes only 5. For the real-world protein-signaling network in Fig. F.2.1, GMP invokes 76580 CIs but C-LMP invokes only 10.

Even when C-LMP invokes exponentially many CIs, LISTCI outputs all such CIs in poly-delay (Thm. 3). This is the first known algorithm that runs in poly-delay where the associated Markov property is applicable to arbitrary data distributions and DAGs with latent variables. The poly-delay property allows researchers to test the subset of CIs listed in the available time, which enables partial testing of the model. This is not possible with an algorithm that takes exponential time to output one CI, or even all CIs at once. Please refer to Appendix A.2 for more details on related literature.

Q6. How well does LISTCI scale?

Answer. LISTCI scales well and is currently the most efficient algorithm that enumerates all CIs invoked by a Markov property which is applicable to arbitrary data distributions and DAGs with latent variables. The plot in Fig. 5 shows that LISTCI takes more than an hour over some graphs with $n >= 70$ nodes. Here, we note that n is not the only factor in the running time of LISTCI. In fact, as shown in Appendix F.3, the graph topology associated with c-components plays a major role in the number of CIs invoked by C-LMP. Two factors related to c-components are of major interest:

(a) $s \leq n$: the size of the largest c-component, and
(b) Sparsity of c-components with respect to the number of bidirected edges

Let **CI** be the total number of non-vacuous CIs invoked by C-LMP. In summary, when c-components are sparse, **CI** increases exponentially in term s, given by the bound $O(n2^s)$. However, as c-components become denser, **CI** decays in exponential term. For illustration, please refer to the discussion on Case 1 in Appendix F.3 (Fig. F.3.1).

Graph topology may vary across different graphs with different sizes. For example, large graphs can have many, small c-components. In this case, s may be small. Then, an exponent s in the bound $O(n2^s)$ is small, and thus total number of CIs invoked by C-LMP may not be large. Even when large graphs have large c-components, if such c-components are dense, then total number of CIs invoked by C-LMP could be smaller in an order of magnitude, as oppose to the case where the c-components are sparse.

Next, there may exist exponentially many CIs invoked by C-LMP (with respect to n), requiring exponential time to list them all. In such cases, one guarantee we can provide is the poly-delay property, which holds for LISTCI (Thm. 3).