Shorter version to appear in AAAI-08.

TECHNICAL REPORT
R-340-L
April 2008

Dormant Independence

Ilya Shpitser and Judea Pearl
Cognitive Systems Laboratory
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA. 90095
$\{ilyas, judea\}@cs.ucla.edu$

Abstract

The construction of causal graphs from non-experimental data rests on a set of constraints that the graph structure imposes on all probability distributions compatible with the graph. These constraints are of two types: conditional independencies and algebraic constraints, first noted by Verma. While conditional independencies are well studied and frequently used in causal induction algorithms, Verma constraints are still poorly understood, and rarely applied. In this paper we examine a special subset of Verma constraints which are easy to understand, easy to identify and easy to apply; they arise from "dormant independencies," namely, conditional independencies that hold in interventional distributions. We give a complete algorithm for determining if a dormant independence between two sets of variables is entailed by the causal graph, such that this independence is identifiable, in other words if it resides in an interventional distribution that can be predicted without resorting to interventions. We further show the usefulness of dormant independencies in model testing and induction by giving an algorithm that uses constraints entailed by dormant independencies to prune extraneous edges from a given causal graph.

Introduction

Graphical causal models [Pearl, 2000] embody both causal and probabilistic assumptions. The vertices in *causal graphs*, the carriers of these assumptions, correspond to variables, while the absence of an edge between two variables implies that those two variables are conditionally independent given some other set of variables. Probabilistic independence between sets of variables in a causal model is implied by the well-known criterion of path blocking called d-separation [Pearl, 1988]. Conversely, independence implies corresponding path blocking in the graph in a special class of models termed *faithful* [Spirtes, Glymour, & Scheines, 1993] or *stable* [Pearl & Verma, 1991], [Pearl, 2000].

Graphs constrain observable distributions in two ways, either by requiring that certain conditional independencies hold, or imposing other restrictions, termed Verma constraints [Verma & Pearl, 1990], [Tian & Pearl, 2002b], which are more difficult to characterize. The constraints induced on the graph by conditional independencies are already being utilized by causal induction algorithms such as **IC** [Verma & Pearl, 1990], [Pearl, 2000], and **FCI** [Spirtes, Glymour, & Scheines, 1993]. A better understanding of Verma constraints may lead to improvements of these algorithms.

In this paper, we examine a special subset of Verma constraints with two nice properties. Firstly, these constraints have a natural interpretation as being due to conditional independencies in distributions resulting from interventions [Pearl, 2000] (we call such independencies *dormant*). Secondly, these constraints have the potential to imply features of the causal graph, specifically the absence of certain edges.

Dormant independencies may imply constraints on the observable distribution, if the interventional distribution in which they reside is *identifiable* [Pearl, 2000], in other words if it can be predicted from observational studies. Our contribution is twofold. We develop a polynomial time algorithm which, given two arbitrary disjoint sets of observable variables, returns an identifiable witness for the dormant independence, in other words an identifiable interventional distribution in which these sets are conditionally independent. Moreover, we show that our algorithm is complete for determining all identifiable dormant independencies entailed by the causal graph, in a sense that if the algorithm fails, then any identifiable dormant independence is "coincidental," and not due to the structure of the graph. Our algorithm is an improvement over a previous algorithm in [Tian & Pearl, 2002b], which enumerated only unconditional dormant independence.

We illustrate the applicability of identifiable dormant independencies for model testing and induction by giving another algorithm which, given a causal graph where every edge is either correct or extraneous (we call such graphs *valid*), uses constraints induced by dormant independencies to systematically rule out extraneous edges.

Our paper is organized as follows. The next section gives an example of a Verma constraint, and shows how this constraint arises due to conditional independence in identifiable interventional distributions. Section 3 goes over the mathematical preliminaries necessary for causal inference. Sections 4 and 5 develop the algorithm for finding witnesses for dormant independence for pairs of singletons, and pairs of arbitrary sets, respectively. Section 6 introduces the algorithm which uses dormant independencies for testing edges. For exposition reasons, some of the longer proofs are found

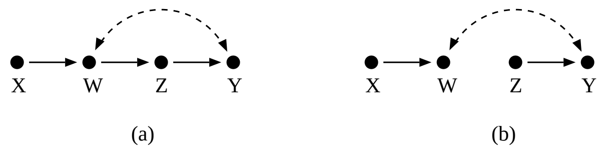

Figure 1: (a) The "P" graph. (b) The graph of the submodel M_z derived from the "P" graph.

in the Appendix.

Verma Constraints as Dormant Independencies

Consider the causal graph in Fig. 1 (a). Any model compatible with this graph imposes certain constraints on its observable distribution $P(x, w, z, y)$. Some of these constraints are in the form of conditional independencies. For instance, in any such model X is independent of Z given W, which means $P(x|w) = P(x|w, z)$. However, there is an additional constraint implied by this graph which cannot be expressed in terms of conditional independence in the observable distribution. This constraint, noted in [Verma & Pearl, 1990], states that the expression $\sum_w P(y|z, w, x)P(w|x)$ is a function of y and z only, not of x. The key insight that motivates this paper is that this constraint does emanate from conditional independencies, albeit not the original observable distribution, but in a distribution resulting from an intervention.

An intervention, written $do(\mathbf{x})$ [Pearl, 2000], is an operation which forces variables $\mathbf{X}$ to attain values $\mathbf{x}$ regardless of their usual behavior in a causal model. The result of applying an intervention $do(\mathbf{x})$ on a model M with a set of observable variables $\mathbf{V}$ is a *submodel* $M_\mathbf{x}$, with stochastic behavior of variables other than $\mathbf{X}$ described by an *interventional distribution* written as $P_\mathbf{x}(\mathbf{v})$ or $P(\mathbf{v}|do(\mathbf{x}))$. The graph induced by $M_\mathbf{x}$ is almost the same as the graph induced by M, except it is missing all arrows incoming to $\mathbf{X}$, to represent the fact that an intervention sets the values of $\mathbf{X}$ independently of its usual causal influences, represented by such arrows. We will denote such a graph as $G_{\overline{\mathbf{x}}}$. Following [Pearl, 2000], we call the set of all possible interventional distributions P_*. In other words, $P_* = \{P_\mathbf{x}(\mathbf{v} \setminus \mathbf{x})|\mathbf{x} \subseteq \mathbf{v}\}$.

A key idea in causal inference is that in certain causal models, some interventional distributions can be predicted or *identified* from the observational distribution. What we will show is that it is ability to identify interventional distributions from observational distributions that gives rise to Verma constraints, including the constraint in the P graph.

Consider a model M inducing the graph in Fig. 1 (a). If we intervene on Z in M, we obtain the submodel M_z inducing the graph in Fig. 1 (b). The distribution of the unfixed observables in this submodel, $P_z(x, w, y)$, is identifiable from $P(x, w, z, y)$ and equals to $P(y|z, w, x)P(w|x)P(x)$ [Tian & Pearl, 2002a]. Moreover, by d-separation [Pearl, 1988], the graph in Fig. 1 (b) implies that X is independent of Y in $P_z(x, w, y)$, or $P_z(y|x) = P_z(y)$. But it's not hard to show that $P_z(y|x)$ is equal to $\sum_w P(y|z, w, x)P(w|x)$, which means this expression depends only on z and y. Thus, the identifiability of $P_z(x, w, y)$ leads to a constraint on observational distributions in the original, unmutilated model M.

Enumerating constraints of this type can be used to infer features of the causal graphs, just as conditional independencies are used for this purpose by causal induction algorithms. For example, establishing that X is independent of Y in $P_z(x, w, y)$ allows us to conclude that the causal graph lacks an edge between X and Y, assuming that the submodel M_z is stable [Pearl & Verma, 1991], [Pearl, 2000], or faithful [Spirtes, Glymour, & Scheines, 1993]. Moreover, since $P_z(x, w, y)$ is identifiable from $P(\mathbf{v})$ in the graph in question, we can rule out the edge without relying on interventions.

In the remainder of this paper, we will show how to achieve a full enumeration of conditional independencies in identifiable interventional distributions entailed by the structure of the graph, and how to use these independencies to infer features of the graph.

Preliminaries

The fundamental object of causal inference is the probabilistic causal model.

Definition 1 *A probabilistic causal model (PCM) is a tuple* $M = \langle \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{F}, P(\boldsymbol{u})\rangle$, *where*

$\boldsymbol{U}$ is a set of background or exogenous variables, which cannot be observed or experimented on, but which can influence the rest of the model.

$\boldsymbol{V}$ is a set $\{V_1, ..., V_n\}$ of observable or endogenous variables. These variables are considered to be functionally dependent on some subset of $\boldsymbol{U} \cup \boldsymbol{V}$.

$\boldsymbol{F}$ is a set of functions $\{f_1, ..., f_n\}$ such that each f_i is a mapping from a subset of $\boldsymbol{U} \cup \boldsymbol{V} \setminus \{V_i\}$ to V_i, and such that $\bigcup \boldsymbol{F}$ is a function from $\boldsymbol{U}$ to $\boldsymbol{V}$.

$P(\boldsymbol{u})$ is a joint probability distribution over the variables in $\boldsymbol{U}$.

PCMs represent causal relationships between observable variables in $\mathbf{V}$ by means of the functions $\mathbf{F}$: a given variable V_i is causally determined by f_i using the values of the variables in the domain of f_i. Causal relationships entailed by a given PCM have an intuitive visual representation using a graph called a causal diagram. As mentioned in the introduction, causal diagrams contain two kinds of edges. Directed edges are drawn from a variable X to a variable V_i if X appears in the domain of f_i. Bidirected edges are always drawn between observable variables, and only when their corresponding functions both make use of the same background variable. In this paper, we consider models which induce acyclic graphs where $P(\mathbf{u}) = \prod_i P(u_i)$, and each U_i has at most two observable children. A graph obtained in this way from a model is said to be induced by said model.

The importance of causal diagrams stems from the fact that conditional independencies between observable variables correspond to graphical features in the diagram. Since the rest of the paper will rely heavily on this correspondence, we introduce probabilistic and graphical notions we will need to make use of it. The key probabilistic notion we will use is the standard definition of condition independence. A set $\mathbf{X}$ is independent of $\mathbf{Y}$ conditional on $\mathbf{Z}$ (written as $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$) if $P(\mathbf{x}|\mathbf{y},\mathbf{z}) = P(\mathbf{x}|\mathbf{z})$. We will use the following graph-theoretic notation. $An(.)_G, De(.)_G, Pa(.)_G$ stand for the set of ancestors, descendants and parents of a given variable set in G. The sets $An(.)_G$ and $De(.)_G$ will be inclusive, in other words, for every $An(X)_G, De(X)_G, X \in An(X)_G$ and $X \in De(X)_G$. A graph $G_{\mathbf{x}}$ stands for the subgraph of G containing only nodes in $\mathbf{X}$ and edges between these nodes. We denote a maximal set of nodes in G pairwise connected by bidirected paths a C-component [Tian, 2002]. We denote the C-component containing a given node X in G by $C(X)_G$. We will drop the graph subscript if the graph is assumed or obvious.

It's possible to show that whenever edges in a causal diagram are drawn according to the above rules, the distribution $P(\mathbf{u},\mathbf{v})$ induced by $P(\mathbf{u})$ and $\mathbf{F}$ factorizes as $\prod_{X_i \in \mathbf{V}\cup\mathbf{U}} P(x_i|Pa(x_i))$. This factorization implies that conditional independencies in $P(\mathbf{u},\mathbf{v})$ are mirrored by a graphical notion of "path blocking" known as d-separation [Pearl, 1988].

Definition 2 (d-separation) *A path p in G is said to be d-separated by a set $\mathbf{Z}$ if and only if either*

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

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

Two sets $\mathbf{X}, \mathbf{Y}$ are said to be d-separated given $\mathbf{Z}$ (written $\mathbf{X} \perp \mathbf{Y}|\mathbf{Z}$) in G if all paths from $\mathbf{X}$ to $\mathbf{Y}$ in G are d-separated by $\mathbf{Z}$. Paths or sets which are not d-separated are said to be d-connected. The relationship between d-separation and conditional independence is provided by the following well-known theorem.

Theorem 1 *Let G be a causal diagram. Then in any model M inducing G, if $\mathbf{X} \perp \mathbf{Y}|\mathbf{Z}$, then $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$.*

Using d-separation as a guide, we can look for a conditioning set $\mathbf{Z}$ which renders given sets $\mathbf{X}$ and $\mathbf{Y}$ independent by only examining the causal diagram, without having to inspect the probability distribution $P(\mathbf{v})$.

In this paper, we examine probabilistic independencies in distributions resulting from not only conditioning but a second, powerful operation of intervention, defined in the previous section. An intervention is a more powerful operation than conditioning, for the purposes of determining probabilistic independence. This is because conditioning on a variable can d-separate certain paths, but also d-connect certain paths (due to the presence of colliders). On the other hand, interventions can only block paths, since incoming arrows are cut by interventions, destroying all colliders involving the intervened variable. Moreover, if we restrict ourselves to interventions identifiable from the observational distribution, we don't pay the price for this power, in a sense that we don't use any information other than the observational distribution, and the causal graph.

Identifiability can be defined formally as follows.

Definition 3 (identifiability) *Consider a class of models $\boldsymbol{M}$ with a description T, and two objects ϕ and θ computable from each model. I say that ϕ is θ-identified in T if ϕ is uniquely computable from θ in any $M \in \boldsymbol{M}$. In other words all models in $\boldsymbol{M}$ which agree on θ will also agree on ϕ.*

If ϕ is θ-identifiable in T, we write $T, \theta \vdash_{id} \phi$. Otherwise, we write $T, \theta \not\vdash_{id} \phi$. In our case, the model class T corresponds to a causal graph, θ is the observational distribution $P(\mathbf{v})$, and ϕ is the causal effect $P(\mathbf{y}|do(\mathbf{x}))$ of interest. For example, in Fig. 1 (a), $P(\mathbf{v}), G \vdash_{id} P_z(x, w, y)$.

We call conditional independencies in interventional distributions *dormant*, to emphasize the fact that such independencies are not apparent in a given observational distribution without causal assumptions that make interventions a meaningful operation.

Definition 4 (dormant independence) *A dormant (conditional) independence exists between variable sets $\boldsymbol{X}, \boldsymbol{Y}$ in $P(\boldsymbol{v})$ obtained from the causal graph G if there exist variable sets $\mathbf{Z}, \boldsymbol{W}$ such that $P(\boldsymbol{y}|\mathbf{x},\mathbf{z},do(\boldsymbol{w})) = P(\boldsymbol{y}|\mathbf{z},do(\boldsymbol{w}))$. Furthermore, if $P(\boldsymbol{v}), G \vdash_{id} P(\boldsymbol{y},\mathbf{x}|\mathbf{z},do(\boldsymbol{w}))$, the dormant independence is identifiable and we denote this as $\boldsymbol{X} \perp\!\!\!\perp_{\boldsymbol{w}} \boldsymbol{Y}|\boldsymbol{Z}$. If an identifiable dormant independence does not exist between $\boldsymbol{X}, \boldsymbol{Y}$ we write $\boldsymbol{X} \not\perp\!\!\!\perp_{*} \boldsymbol{Y}$.*

We can extend the definition of d-separation in a straightforward way to mirror identifiable dormant independencies.

Definition 5 (d*-separation) *Let G be a causal diagram. Variable sets $\boldsymbol{X}, \boldsymbol{Y}$ are d*-separated in G given $\mathbf{Z}, \boldsymbol{W}$ (written $\boldsymbol{X} \perp_{\boldsymbol{w}} \boldsymbol{Y}|\boldsymbol{Z}$), if we can find sets $\mathbf{Z}, \boldsymbol{W}$, such that $\boldsymbol{X} \perp \boldsymbol{Y}|\mathbf{Z}$ in $G_{\overline{\mathbf{w}}}$, and $P(\boldsymbol{v}), G \vdash_{id} P(\boldsymbol{y},\mathbf{x}|\mathbf{z},do(\boldsymbol{w}))$. If $\boldsymbol{X}, \boldsymbol{Y}$ are not d*-separable, we write $\boldsymbol{X} \not\perp_{*} \boldsymbol{Y}$.*

Note that despite the presence of probability notation in the definition, this is a purely graphical notion, since identification can be determined using only the graph. We can prove a theorem analogous to Theorem 1 for dormant independencies, which allows us to reason about dormant independencies graphically.

Theorem 2 *Let G be a causal diagram. Then in any model M inducing G, if $\boldsymbol{X} \perp_{\boldsymbol{w}} \boldsymbol{Y}|\boldsymbol{Z}$, then $\boldsymbol{X} \perp\!\!\!\perp_{\boldsymbol{w}} \boldsymbol{Y}|\boldsymbol{Z}$.*

Proof: This follows from the fact that $G_{\overline{\mathbf{w}}}$ is the graph induced by the submodel $M_{\mathbf{w}}$, and any submodel is just an ordinary causal model where Theorem 1 holds. $\square$

In this paper we seek to characterize cases when arbitrary disjoint sets can be d*-separated, and therefore to characterize identifiable dormant independencies among sets which are entailed by causal graphs. The next section will consider the simpler version of the problem where $\mathbf{X}$ and $\mathbf{Y}$ are singleton sets.

D*-separation Among Singletons

To characterize identifiable dormant independence between X and Y, it makes sense to consider the "difficult" neighborhoods of X, Y, in a sense that no intervention on those neighborhoods is identifiable. We call such neighborhoods ancestral confounding sets.

```
function Find-MACS($G, Y$)
INPUT: $G$, a causal diagram, $Y$ a node in $G$.
OUTPUT: $T_y$, the MACS for $Y$ in $G$.

1 If $(\exists X \notin An(Y)_G)$,
  return Find-MACS($G_{An(Y)}, Y$).
2 If $(\exists X \notin C(Y)_G)$,
  return Find-MACS($G_{C(Y)}, Y$).
3 Else, return $G$.
```

Figure 2: An algorithm for computing the MACS of Y.

Definition 6 *Let Y be a variable in G. A set S is ancestral confounded (ACS) for Y if $S = An(Y)_{G_S} = C(Y)_{G_S}$.*

Ancestral confounded sets are a "difficult" neighborhood due to the following result.

Theorem 3 *Let S be ancestral confounded for Y. Then for any $S' \subseteq S \setminus \{Y\}$, $P(\mathbf{v}), G \not\vdash_{id} P(y|do(s'))$.*

Proof: It's trivial to construct a Y-rooted C-tree T [Shpitser & Pearl, 2006b] from S. But it is known that for any set S' of nodes in T that does not contain Y, $P(\mathbf{v}), G \not\vdash_{id} P(y|do(s'))$ [Shpitser & Pearl, 2006b]. □

In our search for suitable variables to intervene on, in order to separate X and Y, we can exclude ancestral confounded sets for X and Y. But there can be potentially many such sets. It would be preferable to exclude all such sets at once. Fortunately, the following results allows us to accomplish just that.

Theorem 4 *For any variable Y in G, there exists a unique maximum ancestral confounded set (MACS) T_y.*

Proof outline: The key step is to note that if two maximal ancestral confounded sets for Y exist, then their union is also ancestral confounded. □

T_y contains all ancestral confounded sets for Y, which means if we can find an efficient procedure for computing T_y, we could rule out all "difficult" sets from consideration at once. Such an algorithm exists, and is given in Fig. 2.

Theorem 5 ***Find-MACS**(G, Y) outputs the MACS of Y in polynomial time in the size of the graph.*

Proof outline: It's easy to see that the output of **Find-MACS** is an ACS if given a singleton input. To see that it is maximum, we note that **Find-MACS** can never remove elements from T_y at any stage. The algorithm is polynomial since determining $An(.)$ and $C(.)$ sets can be done in polynomial time in the size of the graph, and each recursive call eliminates at least one node from the graph. □

One problem with a MACS T_y is that interventions in T_y are not identifiable, and conditioning in T_y does not d-separate paths from Y out of T_y which consist entirely of colliders, although all paths with a non-collider in T_y are blocked. In order to block some all-collider paths out of T_x, T_y we attempt to intervene on the set $Pa(T_x \cup T_y) \setminus (T_x \cup T_y)$. It turns out these interventions are sufficient to create identifiable dormant independence among singletons, if one exists.

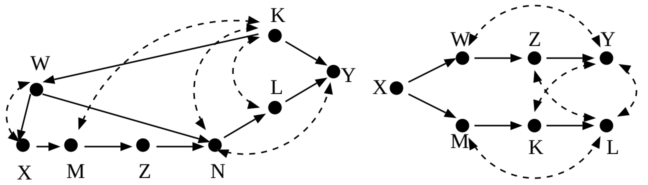

Figure 3: (a) A graph where $X \perp_z Y | W, K, L, N$. (b) A graph where $X \perp_z Y$, $X \perp_k L$, but $X \not\perp_* \{Y, L\}$.

Theorem 6 *Let T_x, T_y be the MACSs of X, Y. Let $I_{x,y} = Pa(T_x \cup T_y) \setminus (T_x \cup T_y)$. Then if either X is a parent of T_y, Y is a parent of T_x or there is a bidirected arc between T_x an T_y, then X, Y are not d*-separable. Otherwise, $X \perp_{i_{x,y}} Y | T_x \cup T_y \setminus \{X, Y\}$.*

Proof outline: If X is a parent of T_y (or vice versa), or there is a bidirected arc between T_x and T_y, then there exists an inducing path between X and Y, which means conditioning cannot separate X and Y. However, we know from results in [Shpitser & Pearl, 2006b] and [Shpitser & Pearl, 2006a] that conditional effects $P_{\mathbf{x}}(\mathbf{y}|\mathbf{z})$ are equivalent to unconditional effects of the form $P_{\mathbf{x'}}(\mathbf{y'})$, and such effects are not identifiable if $Y \in \mathbf{y}'$, and $\mathbf{x}' \subseteq T_y$. This means interventions also don't help.

To show the other direction, note that a d-connected path cannot start with outgoing arrows from both X and Y (else such a path must run into a collider, or an intervened node). Without loss of generality, say the path crosses $T_x \setminus \{X\}$. A directed arrow leaving T_x is blocked either at T_x or $I_{x,y}$, while a bidirected arrow cannot connect to T_y directly, and otherwise must be blocked at $I_{x,y}$. □

To illustrate this theorem, consider the graph in Fig. 3. Here, $T_y = \{K, L, N, Y\}$, and $T_x = \{W, X\}$. By Theorem 6, $X \perp_z Y | W, K, L, N$.

Thus, the MACSs turn out to be key structures for determining identifiable dormant independence between two variables. In the next section, we generalize our results to handle dormant independence among sets of variables.

D*-separation Among Sets

To determine identifiable dormant independencies between sets $\mathbf{X}, \mathbf{Y}$, we want to find a multi-node generalization of MACSs. Unfortunately, we are presented with the following problem. Assume $\mathbf{Y} = \{K, L\}$ such that $K \perp_{\mathbf{w}} L | \mathbf{Z}$. In this case, there really isn't a difficult neighborhood adjacent to both K and L. An appropriate generalization of MACSs to a set $\mathbf{Y}$, then, must partition $\mathbf{Y}$ such that a difficult neighborhood can be defined for each subset in the partition.

Definition 7 *Let $\mathbf{Y}$ be a variable set in G. A set S is ancestral confounded for $\mathbf{Y}$ if for every $Y \in \mathbf{Y}, S = An(\mathbf{Y})_{G_S} = C(Y)_{G_S}$.*

Note that ancestral confounded sets are not guaranteed to exist for sets of nodes. However, if they do exist for some set, finiteness of graphs we consider guarantees the existence of a maximal ancestral confounded set. We want to define an appropriate partition of an arbitrary set, where each element of the partition has an ACS. We will show the following definition will work for this purpose.

Definition 8 (AC-component) *A set* $\mathbf{Y}$ *of nodes in* G *is an ancestral confounded component (AC-component) if*

- $\mathbf{Y} = \{Y\}$, *e.g.,* $\mathbf{Y}$ *is a singleton set, or*
- $\mathbf{Y}$ *is a union of two distinct AC-components* $\mathbf{Y}_1, \mathbf{Y}_2$ *which have ancestral confounded sets* S_1, S_2*, respectively, and* S_1, S_2 *are connected by a bidirected arc*

Lemma 1 *Every AC-component has an ancestral confounded set.*

Proof: If an AC-component is a singleton, this is obvious. Otherwise, $\mathbf{Y}$ is a union of AC-components $\mathbf{Y}_1, \mathbf{Y}_2$ with ancestral confounded sets S_1, S_2. Let $S = S_1 \cup S_2$. Since there is a bidirected arc from S_1 to S_2, for every node $X \in S, S = C(X)_{G_S}$. Moreover, by construction $S = An(\mathbf{Y})_{G_S}$. Thus, S is an ancestral confounded set for $\mathbf{Y}$. □

AC-components behave just as singleton sets do with respect to ACS. In fact, there is a unique MACS for every AC-component, and the algorithm to find it is the familiar **Find-MACS** with set inputs.

Theorem 7 *Let* $\mathbf{Y}$ *be an AC-component. Then there exists a unique MACS* $T_{\mathbf{y}}$ *for* $\mathbf{Y}$*, and* ***Find-MACS-on-set*** *(shown in Fig. 4) finds it in polynomial time in the size of the graph.*

Proof: The proof is a straightforward generalization of the proof of Theorems 4 and 5. □

What we have shown is that certain special sets of nodes have a MACS, just as singletons do. While we cannot show the same for arbitrary sets (consider a set of two nodes not in the same C-component), we can show the next best thing, namely that there exists a unique partition of any set into AC-components.

Lemma 2 *Let* $\mathbf{Y}$ *be a variable set,* $Y \in \mathbf{Y}$*. Then there is a unique maximum AC-component which both contains* Y *and is a subset of* $\mathbf{Y}$*.*

Proof: Some such AC-component exists, since Y itself is a trivial AC-component. Since $\mathbf{Y}$ is finite there is a maximal such AC-component. Assume there are two distinct maximal AC-components containing Y which are subsets of $\mathbf{Y}$, say $\mathbf{Y}_1, \mathbf{Y}_2$. Let S_1, S_2 be the corresponding MACSs. Since these AC-components have the node Y in common, S_1 and S_2 have a node in common, and so are connected by a bidirected arc. This implies $\mathbf{Y}_1 \cup \mathbf{Y}_2$ is an AC-component, which is a contradiction. □

Theorem 8 *Any variable set* $\mathbf{Y}$ *has a unique partition* p*, called the AC-partition, where each element* S *in* p *is a maximal AC-component in a sense that no superset of* S *which is also a subset of* $\mathbf{Y}$ *is an AC-component.*

Proof: To see that there is a unique AC-partition p, start with some node $Y \in \mathbf{Y}$, find it's unique maximum AC-component which is still a subset of $\mathbf{Y}$, and repeat the process for the nodes which have not been made part of some AC-component. The set of AC-components obtained in this way is a partition where each element is a maximal AC-component. Since each AC-component is also maximum and unique, p is unique. □

```
function Find-AC-Partition($G, \mathbf{Y}$)
INPUT: $G$, a causal diagram, $\mathbf{Y}$ a set of nodes in $G$.
OUTPUT: $p$, the unique partition of $\mathbf{Y}$ into AC-components,
and the unique MACS $T_{\mathbf{s}}$ for each $\mathbf{S} \in P$.

1 Let $p$ be the partition of $\mathbf{Y}$ containing all singleton subsets
  of $\mathbf{Y}$.
2 For each $Y \in \mathbf{Y}$, let $T_y$ = Find-MACS($G, Y$).
3 Repeat until no merges are possible: If $\exists \mathbf{Y}_1, \mathbf{Y}_2 \in p$ such
  that $T_{\mathbf{y}_1}, T_{\mathbf{y}_2}$ share a bidirected arc, merge $\mathbf{Y}_1, \mathbf{Y}_2$ into $\mathbf{Y}'$
  in $p$, and let $T_{\mathbf{y}'}$ = Find-MACS-on-set($G, \mathbf{Y}'$).
4 return $p$, and the set of MACSs for each element in $p$.

function Find-MACS-on-set($G, \mathbf{Y}$)
INPUT: $G$, a causal diagram, $\mathbf{Y}$ an AC-component in $G$.
OUTPUT: $T_{\mathbf{y}}$, the MACS for $\mathbf{Y}$ in $G$.

1 If ($\exists X \not\in An(\mathbf{Y})_G$),
  return Find-MACS-on-set($G_{An(\mathbf{Y})}, \mathbf{Y}$).
2 If ($\exists Y \in \mathbf{Y}, \exists X \not\in C(Y)_G$),
  return Find-MACS-on-set($G_{C(Y)}, \mathbf{Y}$).
3 Else, return $G$.
```

Figure 4: An algorithm for computing the AC-partition (and the corresponding sets of MACSs) of $\mathbf{Y}$.

There is a simple algorithm, shown in Fig. 4, which, given an arbitrary set $\mathbf{Y}$, finds the unique AC-partition p of $\mathbf{Y}$, and finds the MACS for each AC-component in p.

Theorem 9 ***Find-AC-Partition****($G, \mathbf{Y}$) outputs the unique AC-partition of* $\mathbf{Y}$*, and the set of MACSs for each element in the partition. Moreover, it does so in time polynomial in the size of* G*.*

Proof outline: The output of **Find-AC-partition** is a partition p of $\mathbf{Y}$ where each element is an AC-component. It's a consequence of Lemma 2 that the AC-partition of $\mathbf{Y}$ is coarser than p. If the AC-partition is not equal to p, it's not difficult to derive a contradiction using the definition of AC-components, and the structure of **Find-AC-Partition**. To see that the algorithm is polynomial, note that each invocation of **Find-MACS** and **Find-MACS-on-set** terminates in polynomial time, as those algorithms are themselves polynomial. Moreover, the set merge operations performed can be easily implemented in polynomial time. Finally, the total number of set merges performed by the algorithm is bounded by the number of nodes in a binary tree with the number of leaves equal to the number of nodes in G. This means the number of merges is linear in the size of G, and the overall algorithm is polynomial in the size of G. □

We want to prove a result analogous to Theorem 6 for sets. To do so, we must generalize the notion of an inducing path to sets.

Definition 9 (inducing paths for sets) *Let $\mathbf{X}, \mathbf{Y}$ be sets of variables in G. A path p between $\mathbf{X}$ and $\mathbf{Y}$ is called an inducing path if the following two conditions hold*

- *The path forms a collider for every non-terminal node*
- *Every non-terminal node is an ancestor of $\mathbf{X}$ or $\mathbf{Y}$.*

Not surprisingly, inducing paths characterize d-separability for sets just as they do for singleton variables.

Theorem 10 *$\mathbf{X}$ cannot be d-separated from $\mathbf{Y}$ in G if and only if there exists an inducing path from $\mathbf{X}$ to $\mathbf{Y}$ in G,*

Proof outline: If there is no inducing path from $\mathbf{X}$ to $\mathbf{Y}$, then $\mathbf{A} = An(\mathbf{X} \cup \mathbf{Y}) \setminus \mathbf{X} \cup \mathbf{Y}$ will serve as a d-separating set. Any path not involving nodes in $\mathbf{A}$ must contain a collider and so isn't d-connecting. Since we condition on $\mathbf{A}$, the d-connecting path must contain only colliders, but this contradicts the absence of an inducing path.

If there is an inducing path, we can establish by case analysis on this path that $\mathbf{X} \not\perp \mathbf{Y}$. The key observation is that regardless of what set of nodes we condition on, it is always possible to recover a path which behaves as an inducing path, which means $\mathbf{X}$ and $\mathbf{Y}$ stay d-connected. □

We can now prove the generalization of Theorem 6 for sets. The idea is to find the AC-partition of $\mathbf{X} \cup \mathbf{Y}$, and generalize the two conditions for d*-separability in Theorem 6 for this AC-partition.

Theorem 11 *Let $\mathbf{X}, \mathbf{Y}$ be arbitrary sets of variables. Let p be the AC-partition of $\mathbf{X} \cup \mathbf{Y}$. Then if either elements of both $\mathbf{X}$ and $\mathbf{Y}$ share a single AC-component in p, or some element of $\mathbf{X}$ is a parent of the MACS of some AC-component containing elements of $\mathbf{Y}$ (or vice versa), then $\mathbf{X}$ cannot be d*-separated from $\mathbf{Y}$. Otherwise, let T_p be the union of all MACSs of elements in p, and let $I_p = Pa(T_p) \setminus T_p$. Then, $\mathbf{X} \perp_{i_p} \mathbf{Y} | T_p \setminus (\mathbf{X} \cup \mathbf{Y})$.*

Proof outline: If the above conditions hold, the inducing path between $\mathbf{X}$ and $\mathbf{Y}$ exists by definition. Thus, conditioning will not help to separate $\mathbf{X}$ and $\mathbf{Y}$. To see that interventions also will not help, we can show by induction on AC-component structure that the effect of any subset of the MACS of any AC-component on that AC-component cannot be identified, which implies, using the results in [Shpitser & Pearl, 2006a] we appealed to in the proof of Theorem 6, that interventions also do not help.

The proof of the other direction follows the same lines as the proof in Theorem 6. □

We conclude this section by noting that just as was the case with conditional independence, identifiable dormant independence among subsets does not entail dormant independence on sets. For example, in the graph shown in Fig. 3 (b), $X \perp_z Y$, $X \perp_k L$, but $X \not\perp_* \{Y, L\}$.

Having given a complete solution to the problem of determining identifiable dormant independence implied by causal graph via d*-separation, we give an example of how such independencies can be used to test aspects of the causal diagram.

Testing Causal Structure

To illustrate the usefulness of identifiable dormant independencies for induction and testing of causal structures, we consider the problem of detecting if certain edges in a particular causal graph are extraneous. We call graphs where every edge is either correct or extraneous valid.

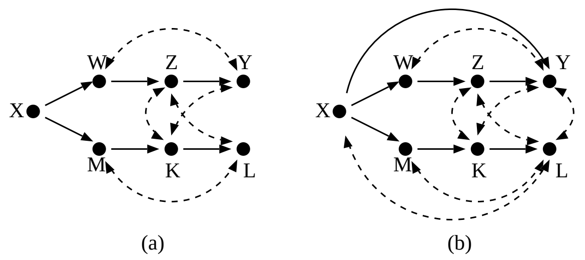

Figure 5: (a) The true causal graph. (b) A possible valid graph for the same domain.

Definition 10 (valid graph) *A causal graph G is valid for a model M if every edge in the graph induced by M is present in G.*

It is possible to rule out out the presence of certain extraneous edges using conditional independence tests. In order to do so, an additional property of *faithfulness* is assumed. In faithful models, lack of d-separation implies dependence. In other words, $\mathbf{X} \perp \mathbf{Y} | \mathbf{Z}$ iff $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$. This property allows us to reach graphical conclusions from probabilistic premises. For instance, the presence of a conditioning set $\mathbf{Z}$ such that $X \perp\!\!\!\perp Y | \mathbf{Z}$ implies X and Y cannot share an edge. Systematic use of conditional independence tests to rule out adjacencies in this way is an important part of causal inference algorithms such as **IC** [Verma & Pearl, 1990], [Pearl, 2000] and **FCI** [Spirtes, Glymour, & Scheines, 1993].

The advantage of dormant independencies is their ability to rule out edges even if all conditional independence tests fail. For instance, it is possible to rule out the edge from X to Y in Fig. 5 (b) as extraneous if $X \perp\!\!\!\perp_z Y$, though no conditional independence test can succeed in doing the same, since there is an inducing path from X to Y.

However, in order to reach graphical conclusions from dormant independencies, we need to extend the faithfulness property to hold in interventional settings.

Definition 11 (experimental faithfulness) *A model M is experimentally faithful, or P_*-faithful if every submodel $M_{\mathbf{x}}$ of M is faithful (that is d-connectedness in $G_{\overline{\mathbf{x}}}$ implies dependence).*

Experimental faithfulness states that no "numerically coincidental independencies" are introduced by interventions. We use dormant independence tests to rule out extraneous edges in valid graphs of experimentally faithful models. To test if an edge between X and Y is extraneous, we must find sets $\mathbf{Z}, \mathbf{W}$ such that $X \perp\!\!\!\perp_{\mathbf{w}} Y | \mathbf{Z}$. A naive brute-force approach to this problem is intractable since we must try all subsets $\mathbf{Z}, \mathbf{W}$. However, if we assume the edge we are testing is absent in the graph, we can use the **Find-MACS** algorithm to propose a dormant independence to test in polynomial time. Since testing this independence does not require we perform any interventions, the test can be performed on the observational distribution alone. There is an additional

```
function Test-Edges($G, P(\mathbf{v})$)
INPUT: $G$, a valid graph of an experimentally faithful model $M$, $P(\mathbf{v})$, a corresponding probability distribution.
OUTPUT: $G'$, a valid graph with some extraneous edges removed.

• Let $\pi$ be a topological order of edges in $G$, where $(X,Y) \prec_\pi (W,Z)$ if $X,Y \in An(\{W,Z\})_G$. Let $G'$ equal $G$.
• For every edge $(X,Y)$ in $\pi$, if we can find sets $\mathbf{Z}, \mathbf{W}$ using Theorem 6 such that $X \perp_{\mathbf{w}} Y|\mathbf{Z}$ in $G' \setminus (X,Y)$, and
  $X \perp\!\!\!\perp_{\mathbf{w}} Y|\mathbf{Z}$ in $P(\mathbf{v}), G'$, remove $(X,Y)$ from $G'$.
• return $G'$.
```

Figure 6: An algorithm for testing edges in valid graphs.

complication, namely that certain edges ancestral to X and Y may themselves be extraneous. This may result in a situation where $X \not\perp_* Y$ if the ancestral extraneous edges are present, while a dormant independence can be established if they are removed. Fortunately, since we restrict ourselves to acyclic graphs, we can establish a topological order among edges based on ancestry, and test for extraneous edges using this order. The resulting algorithm is shown in Fig. 6

It is not difficult to establish that **Test-Edges** is sound.

Theorem 12 ***Test-Edges*** *terminates in polynomial time in the size of the graph, and any edge it removes from G', valid for an experimentally faithful model M, is extraneous.*

Proof: The first claim is simple to establish since all input graphs are acyclic, and using Theorem 7. Let G be the true causal graph. Assume an edge (X,Y) is not extraneous but is removed from G' by **Test-Edges**. Assume sets $\mathbf{Z}, \mathbf{W}$ witness the removal. But $X \perp\!\!\!\perp_{\mathbf{w}} Y|\mathbf{Z}$, and since the submodel $M_{\mathbf{w}}$ of M is faithful, this implies (X,Y) must be extraneous. □

To illustrate the operation of the algorithm, consider the valid graph G' in Fig. 5 (b). If the graph G in Fig. 5 (a) represents the true causal model, **Test-Edges** will be able to remove the edges (X,Y) and (X,L), but not the edge (L,Y). In the case of (X,Y), $X \perp_z Y$ in $G' \setminus (X,Y)$ and the corresponding dormant independence holds since the true model induces G. Similarly, for (X,L), $X \perp_k L$ in $G' \setminus (X,L)$ and the corresponding dormant independence holds. On the other hand, even though (Y,L) is an extraneous edge, **Test-Edges** cannot remove it, since the algorithm cannot establish dormant independence between Y and L, even though $P(y,l|do(z,k))$ is identifiable in the true model. The intuition here is that this identification relies on the absence of the very edge we are trying to test (since $P(y,l|do(z,k))$ is not identifiable in G').

Similarly, if the graph G shown in Fig. 3 (a) is the true causal graph, and the valid graph contains an extra edge from X to Y, **Test-Edges** will be able to remove this edge since $X \perp_z Y|W,K,L,N$ in G, and $P(\mathbf{v}), G' \vdash_{id} P_z(\mathbf{v} \setminus z)$, where G' is G plus any edge from X to Y.

Conclusions

In this paper we consider dormant independencies, that is conditional independencies that surface in interventional distributions. We give a complete algorithm for the problem of determining identifiable dormant independencies entailed by the causal graph, in other words determining if two sets of random variables can be rendered independent by conditioning in some identifiable interventions. We also provide a characterization of graphical structures which prevent identifiable dormant independencies. We have also demonstrated the usefulness of the notion of dormant independencies for testing and induction of causal structure by giving an algorithm which uses constraints entailed by identifiable dormant independencies to remove extraneous edges from causal diagrams.

Straightforward applications of dormant independencies rely on some knowledge of the graph in order to conclude identification. Extending our results to the situations where the underlying graph is not available, for instance in order to do causal induction, is an interesting area of future work.

Acknowledgments

This work was supported in part by the NLM grant #T15 LM07356.

References

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

[2] Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan and Kaufmann, San Mateo.

[3] Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

[4] Shpitser, I., and Pearl, J. 2006a. Identification of conditional interventional distributions. In *Uncertainty in Artificial Intelligence*, volume 22.

[5] Shpitser, I., and Pearl, J. 2006b. Identification of joint interventional distributions in recursive semi-markovian causal models. In *Twenty-First National Conference on Artificial Intelligence*.

[6] Spirtes, P.; Glymour, C.; and Scheines, R. 1993. *Causation, Prediction, and Search*. Springer Verlag, New York.

[7] Tian, J., and Pearl, J. 2002a. A general identification condition for causal effects. In *Eighteenth National Conference on Artificial Intelligence*, 567–573.

[8] Tian, J., and Pearl, J. 2002b. On the testable implications of causal models with hidden variables. In *Proceedings of UAI-02*, 519–527.

[9] Tian, J. 2002. *Studies in Causal Reasoning and Learning*. Ph.D. Dissertation, Department of Computer Science, University of California, Los Angeles.

[10] Verma, T. S., and Pearl, J. 1990. Equivalence and synthesis of causal models. Technical Report R-150, Department of Computer Science, University of California, Los Angeles.

Appendix

Theorem 4 *For any variable Y in G, there exists a unique maximum ancestral confounded set (MACS) T_y.*

Proof: Maximal ancestral confounded sets exist for any Y since we only consider finite graphs. Assume there is Y with two distinct maximal ancestral confounded sets S_1, S_2. We claim that $S = S_1 \cup S_2$ is an ancestral confounded set, which is a contradiction. By construction, S is a C-component in G_S, since any node $X \in S_1$ and any node $Z \in S_2$ can be connected by a bidirected path constructed by appending the bidirected path from X to Y in G_{S_1} (guaranteed to exist since S_1 is a C-component in G_{S_1}) to the bidirected path from Z to Y in G_{S_2} (guaranteed to exist since S_2 is a C-component in G_{S_2}). Since $S_1 \in An(Y)_{S_1}$, and $S_2 \in An(Y)_{S_2}$, $S \in An(Y)_S$. □

Theorem 5 ***Find-MACS**$(G, \{Y\})$ outputs T_y, the MACS of Y in polynomial time in the size of the graph.*

Proof: The algorithm is polynomial since determining $An(.)$ and $C(.)$ sets can be done in polynomial time, and each recursive call eliminates at least one node from the graph. Since the MACS of Y is unique, all ancestral confounding sets of Y are contained in it (otherwise, we can repeat the argument in Theorem 4). First, we show that the output set S of **Find-MACS** is an ancestral confounding set of Y. If not, then either $S \neq An(Y)_{G_S}$ or $S \neq C(Y)_{G_S}$. But the algorithm only returns if there is no element in S outside $An(Y)_{G_S}$, and no element in S outside $C(Y)_{G_S}$. To show that S is maximum, assume this isn't the case, and let $\mathbf{Z} \subseteq T_y \setminus S$ be the first node set in T_y removed by **Find-MACS**. Let G' be the graph at the stage where $\mathbf{Z}$ is removed. By assumption, T_y is contained in G', and either $\mathbf{Z} \not\subset An(Y)_{G'}$ or $\mathbf{Z} \not\subset C(Y)_{G'}$. But $\mathbf{Z} \subset An(Y)_{G_{T_y}}$, and $\mathbf{Z} \subset C(Y)_{G_{T_y}}$ by definition of T_y. Contradiction. □

Theorem 6 *Let T_x, T_y be the MACSs of X, Y. Let $I_{x,y} = Pa(T_x \cup T_y) \setminus (T_x \cup T_y)$. Then if either X is a parent of T_y, Y is a parent of T_x or there is a bidirected arc between T_x an T_y, then X, Y are not d*-separable. Otherwise, $X \perp_{i_{x,y}} Y | T_x \cup T_y \setminus \{X, Y\}$.*

Proof: Assume either X is a parent of T_y or T_x, T_y are connected by a bidirected arc. It's easy to verify, by definition of T_y, that the the above imply the presence of an inducing path [Verma & Pearl, 1990] from X to Y. Thus, no conditioning set can d-separate X and Y. We want to show that identifiable interventions don't help. Consider disjoint subsets S, S' of T_y. A result in [Shpitser & Pearl, 2006a] implies that $P(\mathbf{v}), G \not\vdash_{id} P(y|s', do(s))$ iff $P(\mathbf{v}), G \not\vdash_{id} P(y, t|do(s, t'))$, where T, T' is a certain partition of S'. By Theorem 3, $P(\mathbf{v}), G \not\vdash_{id} P(y|do(\mathbf{w}))$ for any subset $\mathbf{W}$ of T_y, which in turn implies $P(\mathbf{v}), G \not\vdash_{id} P(y, t|do(s, t))$. But if $P(\mathbf{v}), G \not\vdash_{id} P(y|s', do(s))$, then $P(\mathbf{v}), G \not\vdash_{id} P(y, x|s', do(s))$. It is not difficult to construct a model where for any superset $\mathbf{Z}$ of S', and superset $\mathbf{W}$ of S, $P(\mathbf{v}), G \not\vdash_{id} P(y, x|\mathbf{z}, do(\mathbf{w}))$ (by for instance letting nodes outside T_y be mutually independent). This implies our result.

To show the other direction, consider $G_{\overline{i_{x,y}}}$, and a possible d-connected path from X to Y. This path starts with an arrow leaving X or an arrow entering X. Assume the arrow is leaving X. X cannot have conditioned descendants in $G_{\overline{i_{x,y}}}$ unless X was a parent of T_y or $x \in T_y$, both of which are impossible by assumption. This means the path from X is just a set of directed arrows from X. But such a path must run into nodes fixed by $I_{x,y}$, unless X was a parent of T_y or in T_y, which is impossible. Thus, no path starting with an outgoing arrow from X can be d-connected to Y.

Assume the path starts with an incoming arrow into X. If the arrow is directed, the corresponding parent Z of X is either in T_x or in $I_{x,y}$ (and in neither case can Z be equal to Y). In either case, the path is not d-connected to Y. If the arrow is bidirected, we have two cases. Either the next node Z in the path is in T_y or outside both T_y and $I_{x,y}$ (Z cannot be in $I_{x,y}$ since then the path will not be d-connected). For the first case, we repeat the argument until we reach the second case. For the second case, Z cannot be in T_x, else there is a bidirected path from T_x to T_y, which is ruled out by assumption. Note that Z cannot have conditioned descendants in $G_{\overline{i_{x,y}}}$ unless Z was a parent of T_x or T_y or was in T_x or T_y. But we ruled all these cases out. Therefore, the subsequent arrows on the path are directed arrows away from Z. As before, these arrows must eventually reach $I_{x,y}$, which means the path is not d-connected. □

Theorem 9 ***Find-AC-Partition**$(G, \mathbf{Y})$ outputs the unique AC-partition of $\mathbf{Y}$, and the set of MACSs for each element in p. Moreover, it does so in time polynomial in the size of G.*

Proof: We first show that p, the output of **Find-AC-Partition**, consists of a partition of AC-components (not necessarily maximal). Clearly this is true at the initialization step, since a singleton is a trivial AC-component. It's also clear by definition that any merge of $\mathbf{Y}_1, \mathbf{Y}_2$ results in an AC-component $\mathbf{Y}'$. Furthermore, by Theorem 7, $T_{\mathbf{y}'}$ is the MACS of $\mathbf{Y}'$.

Let p^* be the AC-partition of $\mathbf{Y}$. We claim that p^* must be coarser than p, in a sense that every element in p^* is a union of a set of elements in p. Note that this definition holds if p^* is equal to p. Assume not. Then there are some sets $S \in p, S' \in p^*$ such that some elements in S are in S' and some are not. Let $Z \in S \cap S'$. By Lemma 2, there is a unique maximum AC-component containing Z which is also a subset of $\mathbf{Y}$. By definition of p^*, S' is this AC-component. But if S is not contained in S', we can derive a contradiction by repeating the argument in the proof of Lemma 2.

Finally, we want to show p^* is equal to p. Assume this isn't the case, and fix some element S' in p^* which is a union of two or more elements in p. Since each AC-component is either a singleton, or constructed from two smaller AC-components, we can construct a binary tree T, where each leaf is a node in S', and each non-leaf represents an AC-component obtained from the AC-component corresponding to the left subtree of the non-leaf and the AC-component corresponding to the right subtree of the non-leaf.

We want to find an AC-component A in T with the property that its left subtree corresponds to a subset of some element S_1 in p, and its right subtree corresponds to a subset of another element S_2 in p. This AC-component must exist, since leaves in T are singletons, and the root of T corresponds to S', which spans multiple elements in p. This implies that the MACS of a subset of S_1 is connected to the MACS of a subset of S_2 by a bidirected arc. But the MACS of S_1 and the MACS of S_2 are supersets of these connected MACS, so they are themselves connected by a bidirected arc. But then p could not have been the output of **Find-AC-Partition**. $\square$

Theorem 10 *$\mathbf{X}$ cannot be d-separated from $\mathbf{Y}$ in G if and only if there exists an inducing path from $\mathbf{X}$ to $\mathbf{Y}$ in G,*

Proof: Assume there is no inducing path from $\mathbf{X}$ to $\mathbf{Y}$. Let $\mathbf{A} = An(\mathbf{X} \cup \mathbf{Y}) \setminus (\mathbf{X} \cup \mathbf{Y})$. We claim that $\mathbf{X} \perp \mathbf{Y} | \mathbf{A}$. It's not hard to see that if there is a d-connected path from $\mathbf{X}$ to $\mathbf{Y}$, then it does not have any nodes not in $\mathbf{A}$. Assume otherwise. Then some node on this path not in $\mathbf{A}$ must contain a collider. But this implies the path is not d-connected, since this node does not have descendants in $\mathbf{A}$.

Since we condition on $\mathbf{A}$, the d-connected path must consist exclusively of colliders. Moreover, by definition every node on the path is an ancestor of either $\mathbf{X}$ or $\mathbf{Y}$. But this means the path is inducing. Contradiction.

Assume the inducing path from X to Y. We want to show we cannot d-separate $\mathbf{X}$ from $\mathbf{Y}$. First, we show that $\mathbf{X} \not\perp \mathbf{Y}$.

We have three cases. The inducing path contains either entirely bidirected arcs, or one directed arc following by zero or more bidirected arcs, or one directed arc, following by zero or more bidirected arcs, followed by a directed arc.

Let A be the first node on the inducing path after X, B be the first node on the inducing path after Y. If all nodes on the inducing path are ancestors of $\mathbf{X}$, then B is an ancestor of $\mathbf{X}$. But the edge between Y and B is either bidirected, or directed from Y to B. In either case, the ancestral path from $\mathbf{X}$ to B plus this edge forms a d-connected path from $\mathbf{X}$ to Y. The same argument applies if all nodes on the inducing path are ancestors of $\mathbf{Y}$. Otherwise, find two neighboring nodes C, D on the inducing path where C is an ancestor of $\mathbf{X}$, and D is an ancestor of $\mathbf{Y}$. Then the ancestral path from $\mathbf{X}$ to C, along with the edge along the inducing path from C to D, along with the ancestral path from $\mathbf{Y}$ to D form a d-connected path from $\mathbf{X}$ to $\mathbf{Y}$.

What we have to show is that regardless of which sets of nodes we condition on, some d-connected path between $\mathbf{X}$ and $\mathbf{Y}$ remains. Let p' the subpath of p such that nodes on p' are either conditioned on themselves, or their descendants are conditioned on. If $p' = p$, we are done since p is d-connected. Otherwise, consider every pair of nodes A, B on $p \setminus p'$ such that all nodes on p between A and B are in p'. By construction, the fragment of p between A and B is a d-connected path, terminating with arrowheads on both ends. To show that there is a d-connected path between $\mathbf{X}$ and $\mathbf{Y}$, we repeat the above d-connection argument, except rather than considering the path p, we consider the path $p \setminus p'$, and instead of the d-connected paths between every node pair A, B as above, we consider a bidirected arc. $\square$

Theorem 11 *Let $\boldsymbol{X}, \boldsymbol{Y}$ be arbitrary sets of variables. Let p be the AC-partition of $\boldsymbol{X} \cup \boldsymbol{Y}$. Then if either elements of both $\boldsymbol{X}$ and $\boldsymbol{Y}$ share a single AC-component in p, or some element of $\boldsymbol{X}$ is a parent of the MACS of some AC-component containing elements of $\boldsymbol{Y}$ (or vice versa), then $\boldsymbol{X}$ cannot be d*-separated from $\boldsymbol{Y}$. Otherwise, let T_p be the union of all MACSs of elements in p, and let $I_p = Pa(T_p) \setminus T_p$. Then, $\boldsymbol{X} \perp_{i_p} \boldsymbol{Y} | T_p \setminus (\boldsymbol{X} \cup \boldsymbol{Y})$.*

Proof: What we want to show is that the conditions for the absence of d*-separation of sets $\mathbf{X}, \mathbf{Y}$ imply that there is an inducing path between $\mathbf{X}$ and $\mathbf{Y}$, and that no interventions on nodes in that inducing path are identifiable, at least if either $\mathbf{X}$ or $\mathbf{Y}$ are the effect variables.

We first want to show that if $\mathbf{Z}$ is an AC-component, then for any disjoint subsets S, S' of the MACS $T_{\mathbf{z}}$, $P(\mathbf{v}), G \not\vdash_{id} P(\mathbf{z}|s', do(s))$. By a result from [Shpitser & Pearl, 2006a], $P(\mathbf{v}), G \not\vdash_{id} P(\mathbf{z}|s', do(s))$ iff $P(\mathbf{v}), G \not\vdash_{id} P(\mathbf{z}, t|do(s, t'))$, where T, T' is a particular partition of S'. But if $P(\mathbf{v}), G \not\vdash_{id} P(\mathbf{z}|do(s, t'))$, then $P(\mathbf{v}), G \not\vdash_{id} P(\mathbf{z}, t|do(s, t'))$. Without loss of generality, then, we will prove that $P(\mathbf{v}), G \not\vdash_{id} P(\mathbf{z}|do(s))$. By Theorem 3, this is true if $\mathbf{Z} = \{Z\}$. Assume this is true for AC-components $\mathbf{Z}_1, \mathbf{Z}_2$. We want to show this also holds for the AC-component $\mathbf{Z}$ obtained from these two AC-components. Clearly, the result also holds for $T = T_{\mathbf{z}_1} \cup T_{\mathbf{z}_2}$. We want to show the same is true for $T_{\mathbf{z}}$. By construction, $T_{\mathbf{z}}$ can be used to construct a C-forest [Shpitser & Pearl, 2006b] for $\mathbf{Z}$. The same is true for T. Then $T, T_{\mathbf{z}}$ form a hedge [Shpitser & Pearl, 2006b] for $P(\mathbf{z}|do(s'))$, for any set $S' \subseteq T_{\mathbf{z}} \setminus T$, which means the result holds for $T_{\mathbf{z}}$.

If there is an AC-component containing both elements of $\mathbf{X}$ and $\mathbf{Y}$, then an inducing path between $\mathbf{X}$ and $\mathbf{Y}$ exists by the definition of AC-component. Similarly, if some element of $\mathbf{X}$ is a parent of the MACS of some AC-component which is a subset of $\mathbf{Y}$, then an inducing path between $\mathbf{X}$ and $\mathbf{Y}$ exists by the definition of AC-component.

If there is an AC-component $\mathbf{C}$ containing both elements of $\mathbf{X}$ and $\mathbf{Y}$, then by above reasoning for any disjoint subsets S, S' of $T_{\mathbf{z}}$, $P(\mathbf{v}), G \not\vdash_{id} P(\mathbf{c}|s', do(s))$. Similarly, if there is an element of $\mathbf{X}$ which is a parent of the MACS of some AC-component $\mathbf{Y}'$ which is a subset of $\mathbf{Y}$, then by above reasoning for any disjoint subsets S, S', $P(\mathbf{v}), G \not\vdash_{id} P(\mathbf{y}'|s', do(s))$. As before, it is not difficult to construct a model where for any superset $\mathbf{Z}$ of S' and superset $\mathbf{W}$ of S, $P(\mathbf{v}), G \not\vdash_{id} P(\mathbf{c}|\mathbf{z}, do(\mathbf{w}))$ (in the first case), or $P(\mathbf{v}), G \not\vdash_{id} P(\mathbf{y}'|\mathbf{z}, do(\mathbf{w}))$, (in the second case). In either case, no combination of fixing and conditioning can rid us of the inducing path, and our result follows.

To prove the other direction, consider a d-connected path in $G_{\overline{i_p}}$ from $X \in \mathbf{X}$ to $Y \in \mathbf{Y}$. Without loss of generality, assume no elements in $\mathbf{X}, \mathbf{Y}$, other than the end points are on this path.

The path either starts with an outgoing arrow, an incoming arrow, or a bidirected arrow. Assume it starts with an outgoing arrow into a node Z. If Z is inside some MACS, the next edge on the path can be assumed to be bidirected. This is because this MACS cannot contain any nodes in $\mathbf{Y}$, and because the next node Z is conditioned on by assumption. Since the arrow is bidirected, we handle this case in the "bidirected arrow" situation. If Z is outside any MACS, it is either in I_p, in which case the path is not d-connected, or it does not have any conditioned descendants, since the parents of every MACS are fixed. This means the segment of the path from Z is just a set of directed arrows pointing away from Z. But such a path must run into nodes fixed by

I_p, which is impossible. Thus there are no d-connected path starting with an outgoing arrow from X.

Assume the path starts with an incoming arrow into X. If the arrow is directed, the corresponding parent Z of X is either in T_p. or in I_p. If it is in I_p, the path is not d-connected, since no element of Y can be a parent of the MACS of an AC-component containing X by assumption. If it is in T_p, it is conditioned on, and the path is not d-connected.

If the arrow is bidirected, we have two cases. Either the next node Z in the path is in the MACS of an AC-component containing X, or outside both this AC-component, and I_p. For the first case, we repeat the argument until we reach the second case. For the second case, Z cannot be in any other MACS. Otherwise, there is a bidirected arc between distinct MACSs returned by **Find-AC-Partition** which is impossible by Theorem 9. Note that Z cannot have conditioned descendants in $G_{\overline{i_p}}$ unless Z was in I_p, which is impossible. Therefore, the subsequent arrows on the path are directed arrows away from Z. As before, these arrows must eventually reach I_p, which means the path is not d-connected. $\square$