Uncertainty in Artificial Intelligence 4
R.D. Shachter, T.S. Levitt, L.N. Kanal, J.F. Lemmer (Editors)

TECHNICAL REPORT
R-65
June 1987

CAUSAL NETWORKS: SEMANTICS AND EXPRESSIVENESS*

Thomas VERMA and Judea PEARL

Cognitive Systems Laboratory
Computer Science Department
University of California Los Angeles
Los Angeles, CA 90024
Internet: verma@cs.ucla.edu, pearl@cs.ucla.edu

Dependency knowledge of the form "x is independent of y once z is known" invariably obeys the four *graphoid* axioms, examples include probabilistic and database dependencies. Often, such knowledge can be represented efficiently with graphical structures such as undirected graphs and directed acyclic graphs (DAGs). In this paper we show that the graphical criterion called *d-separation* is a sound rule for reading independencies from any DAG based on a *causal input list* drawn from a graphoid. The rule may be extended to cover DAGs that represent functional dependencies as well as conditional dependencies.

1. INTRODUCTION

In several areas of research it is beneficial to reason about dependency knowledge. For example, in database design it is useful to reason about embedded-multivalued-dependence (EMVD) of attributes [2]. Similarly, in decision analysis and expert systems design it is useful to reason about probabilistic independence of variables [5,9]. These examples give two well known formalizations of the intuitive relation "knowing Z renders X and Y independent" which shall be denoted $I(X,Z,Y)$. Naturally, such a relation would have different properties for different formalizations, but it is interesting to note that most sensible definitions share the four common properties listed below:

symmetry	$I(X,Z,Y) \Leftrightarrow I(Y,Z,X)$	(1.a)
decomposition	$I(X,Z,YW) \Rightarrow I(X,Z,Y)$	(1.b)
weak union	$I(X,Z,YW) \Rightarrow I(X,ZY,W)$	(1.c)
contraction	$I(X,ZY,W)\ \&\ I(X,Z,Y) \Rightarrow I(X,Z,YW)$	(1.d)

where X, Y and Z represent three disjoint subsets of objects (e.g. variables or attributes)

* This work was partially supported by the National Science Foundation Grant #IRI-8610155. "Graphoids: A Computer Representation for Dependencies and Relevance in Automated Reasoning (Computer Information Science)."

and the notation YW is a shorthand for $Y \cup W$. These four properties, in addition to others, hold for EMVDs[1] as well as for probabilistic dependencies [1]. Three place relations which obey the four properties listed above are called *graphoids*[2]. Those relations which obey the following additional axiom:

$$\text{intersection} \quad I(X,ZY,W) \,\&\, I(X,ZW,Y) \Rightarrow I(X,Z,YW) \qquad (2)$$

are called *positive graphoids* since this axiom holds for probabilistic dependencies when the distributions are constrained to be strictly positive (non-extreme distributions).

A naive approach for representing a dependency model, i.e., particular instance of a dependency relation, would be to enumerate all triplets (X,Z,Y) for which $I(X,Z,Y)$ holds. This could require an exponential amount space since the relation I ranges over subsets of objects. In fact, any general representation scheme will require exponential space, on average, to represent a dependency model given that it is a graphoid, probabilistic dependency or EMVD [14].

The graphical representation schemes presented in this paper are appealing for three reasons. First, the graphs have an intuitive conceptual meaning. Second, the representations are efficient in terms of time and space [11]. Third, there exist various efficient algorithms that make implicit use of these representation schemes [6,7,8,9,12,13].

2. UNDIRECTED GRAPHS

The meaning of a particular undirected graph is straight forward, each node in the graph represents a variable, and a link in the graph means that the two variables are directly dependent. With this semantics, a set of nodes Z would *separate* two other sets X and Y, if and only if every path between a node in X and a node in Y passes through Z. This representation can fully represent only a small set of dependency models defined by the following properties [10]:

symmetry	$I(X,Z,Y) \Leftrightarrow I(Y,Z,X)$	(3.a)
decomposition	$I(X,Z,YW) \Rightarrow I(X,Z,Y)$	(3.b)
strong union	$I(X,Z,Y) \Rightarrow I(X,ZW,Y)$	(3.c)
intersection	$I(X,ZY,W) \,\&\, I(X,ZW,Y) \Rightarrow I(X,Z,YW)$	(3.d)
transitivity	$I(X,Z,Y) \Rightarrow I(X,Z,\gamma)$ *or* $I(Y,Z,\gamma)\ \forall\gamma \notin XYZ$	(3.e)

It is not always necessary nor feasible to have an exact representation of a dependency model; in fact, an efficient approximation called an *I-map* is often preferred to an inefficient perfect map. A representation R is an I-map of a dependency model M iff every independence statement represented by R is also a valid independence of M. Thus, R may not represent every statement of M, but the ones it does represent are correct. The rationale behind the use of I-maps is that an ignored independence will cause a redundant consultation thus only cost some time whereas use of an incorrect independence will cause

[1] The notation $I(X,Z,Y)$ is equivalent to the standard EMVD notation $Z \twoheadrightarrow X \mid Y$.

[2] Historically the term *semi-graphoids* was used for graphoids, and *graphoids* was used for positive graphoids.

relevant information to be ignored thereby corrupting the integrity of the system. (This rationale presupposes a specific type of use for the dependency knowledge, thus will not apply in general).

The set of undirected graphs that are I-maps of a given positive graphoid forms a complete lattice under the I-mapness relation. This means that every positive graphoid has a unique most representative I-map graph. Furthermore there is a polynomial algorithm which will find it [10]. Thus, for example, every non-extreme probability distribution P has a unique *edge-minimal* undirected graph I-map of P (i.e. no edge can be removed without destroying the I-mapness of this graph).

This is not the case for EMVD relations nor for probabilistic distributions in general. In fact, with out the intersection property there is no unique edge-minimal I-map for a given model, and, moreover, there is no effective method of constructing even one of the minimal I-maps. Hence undirected graphs are only useful as I-maps of positive graphoids.

3. DIRECTED-ACYCLIC GRAPHS (DAGS)

The dependency model represented by a particular DAG has a simple causal interpretation. Each node represents a variable and there is a directed arc from one node to another if the first is a direct cause of the second. Under this interpretation, graph-separation is not as straight forward as before since two unrelated causes of a symptom may become related once the symptom is observed [8]. In general, a set of nodes Z is defined to *d-separate* two other sets X and Y if and only if every *trail* from a node in X to a node in Y is rendered *inactive* by Z. A trail is a path which follows arcs ignoring their directionality, and is rendered inactive by a set of nodes Z in exactly two ways; either there is a *head-to-head* node on the trail which is not in Z and none of its descendents are in Z or some node on the trail is in Z but is not head-to-head. A node on the trail is head-to-head if the node before it and after it on the trail both point to it in the graph. One node is a descendent of another if there is a directed trail from the latter to the former.

There is a procedure that produces an edge-minimal I-map DAG for any graphoid. It employs an algorithm which takes a *causal input list* (or simply causal list) of a dependency model and produces a perfect map of the list's graphoid closure. A casual list of a dependency model contains two things: an ordering of the variables, and a function that assigns a *tail boundary* to each variable x. For each variable x let U_x denote the set of all variables which come before x in the given ordering. A tail boundary of a variable x, denoted B_x, is any subset of U_x that renders x independent of $U_x - B_x$. A unique DAG can be generated from each casual list by associating the tail boundary of the variable x in the list with the set of direct parents of any node x in the DAG. An equivalent specification of a casual list is an ordered list of triplets of the form $I(x, B_x, R)$, one triplet for each variable in the model, where R is $U_x - B_x$.

For a particular dependency model over n variables there are $n!$ orderings, and for each ordering there can be up to $2^{n(n-1)/2}$ different sets of tail boundaries since, in the worst case, every subset of lesser variables could be a boundary. Thus, there can be as many as $n!\,2^{n(n-1)/2}$ casual lists for any given dependency model. But if a model posses a perfect map DAG, then one of the causal lists is guaranteed to generate that DAG by the following theorem.

Theorem 1: If M is a dependency model which can be perfectly represented by some DAG D, then there is a causal list L_θ which generates D.

Proof: Let D be a DAG which perfectly represents M. Since D is a directed acyclic graph it imposes a partial order ϕ on the variables of M. Let θ be any total ordering consistent with ϕ (i.e. $a <_\phi b \Rightarrow a <_\theta b$). For any node x in D, the set of its parents $P(x)$ constitutes a tail boundary with respect to the ordering θ, thus the pair $L_\theta = (\theta, P(x))$ is a causal list of M, and this is the very list which will generate D. QED.

Although it is possible to find a perfect map when it exists, testing for existence may be intractable. However, it is practical to find an edge-minimal I-map. The next theorem shows that any causal list of a graphoid can be used to generate an I-map of that graphoid. If the boundaries of a causal list are all subset minimal, then the graph it generates will be edge-minimal. Finding a minimal boundary is linear (in the number of variables) due to the weak union property. Hence an edge-minimal I-map of any graphoid can be found in polynomial time.

Theorem 2: If M is a graphoid, and L_θ is any causal list of M, then the DAG generated by L_θ is an I-map of M.

Proof: Induct on the number of variables in the graphoid. For graphoids of one variable it is obvious that the DAG generated is an I-map. Suppose for graphoids with fewer than k variables that the DAG is also an I-map. Let M have k variables, n be the last variable in the ordering θ, $M-n$ be the graphoid formed by removing n and all triplets involving n from M and $G-n$ be the DAG formed by removing n and all its incident links from G. Since n is the last variable in the ordering, it cannot appear in any of boundaries of L_θ, and thus $L_\theta - n$ can be defined to contain only the first $n-1$ variables and boundaries of L_θ and still be a causal list of $M-n$. In fact the DAG generated from $L_\theta - n$ is $G-n$. Since $M-n$ has $k-1$ variables, $G-n$ is an I-map of it. Let M_G be the dependency model corresponding to the DAG G, and M_{G-n} correspond to $G-n$, (i.e. M_G contains all d-separated triplets of G).

G is an I-map of M if and only if $M_G \subseteq M$. Each triplet T of M_G falls into one of four categories; either the variable n does not appear in T or it appears in the first, second or third entry of T. These will be treated separately as cases 1, 2, 3 and 4, respectively.

case-1: If n does not appear in T then T must equal (X, Z, Y) with X, Y and Z three disjoint subsets of variables, none of which contain n. Since T is in M_G it must also be in M_{G-n} for if it were not then there would be an active path in $G-n$ between a node in X and a node in Y when Z is instantiated. But if this path is active in $G-n$ then it must also be active in G since the addition of nodes and links can not deactivate a path. Since $G-n$ is an I-map of $M-n$, T must also be an element of it, but $M-n$ is a subset of M, so T is in M

case-2: If n appears in the first entry of the triplet, then $T = (Xn, Z, Y)$ with the same constraints on X, Y and Z as in case-1. Let (n, B, R) be the last triple in L_θ, B_X, B_Y, B_Z and B_0 be a partitioning of B and R_X, R_Y, R_Z and R_0 be a partitioning of R such that $X = B_X \cup R_X$, $Y = B_Y \cup R_Y$ and $Z = B_Z \cup R_Z$ as in figure 1.

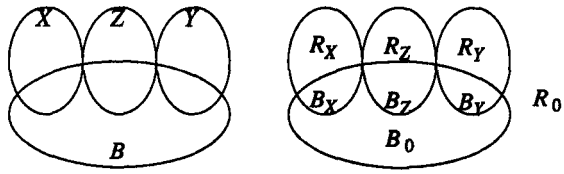

Figure 1

By the method of construction, there is an arrow from every node in B to n, but since (Xn, Z, Y) is in M_G every path from a node in Y to n must be deactivated by Z so B_Y must be empty or else there would be a direct link from Y to n (see figure 2a). The last triplet in L_θ can now be written as $(n, B_X B_0 B_Z, R_X R_Z Y R_0)$. Since $X = B_X \cup R_X$, $Y = R_Y$ and M is a graphoid it follows (from (1.b) and (1.c)) that $(n, XB_0Z, Y) \in M$.

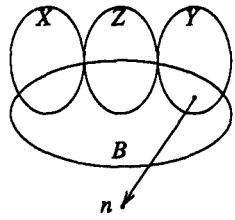

Figure 2a

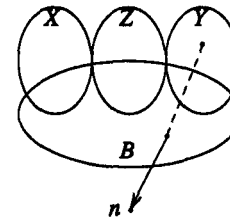

Figure 2b

Since there is an arrow from every node in B_0 to n and n is separated from Y given Z in G, B_0 must also be d-separated from Y given Z in G for if it were connected there would be a path from a node in Y to a node in B_0 which was active given Z. But there is an arrow from every node in B_0 to n, thus, such a path would also connect the node in Y to n, and Y would no longer be separated from n given Z (see figure 2b). Since Y is separated from both B_0 and X given Z in the DAG G it is separated from their union, so $(XB_0, Z, Y) \in M_G$. Since n is not in this triplet, the argument of case-1 above implies that $(XB_0, Z, Y) \in M$. Since $(n, XB_0Z, Y) \in M$ and M is a graphoid it follows (using (1.b) and (1.d)) that $T = (Xn, Z, Y) \in M$

case-3: If n appears in the second entry then $T = (X, Zn, Y)$. Now it must be the case that X and Y are separated in G given only Z for if they were not then there would be a path between some node in X and some node in Y which would be active given Z. But they are separated given Z and n, so this path would have to be deactivated by n, but n is a sink and cannot serve to deactivate any path by being instantiated. Hence there is no such path and (X, Z, Y) holds in G. This statement along with (X, Zn, Y) imply that either (Xn, Z, Y) or (X, Z, Yn) holds in G by the weak transitivity property of DAGs. Either way

case-2 or case-4 would imply that the corresponding triplet must be in M, and either of these would imply that $T \in M$ by the weak union property.

case-4: If n appears in the third entry, then by symmetry the triplet T is equivalent to one with n in the first entry, and the argument of case-2 above shows that $T \in M$. QED.

This theorem constructively proves that d-separation is sound. In other words a DAG built from a causal list is an I-map since for any independence identified by d-separation there exists a derivation of that statement from the the causal input list using the graphoid axioms. The next corollary shows that for DAGs build from causal lists of graphoids, no criterion could correctly read more independencies than d-separation.

Corollary 1: If L_θ is any causal list of some dependency model M, the DAG generated from L_θ is a perfect map of the graphoid closure of L_θ. In other words, a triplet is d-separated in the DAG if and only if it can be derived from the causal input list using the graphoid axioms.

Proof: By the previous theorem, the DAG is an I-map of the closure, and it remains to show that the closure is an I-map of the DAG. Since every DAG dependency model is a graphoid, the DAG closure of L_θ contains the graphoid closure of it, thus, it suffices to show that the DAG dependency model M_G contains L_θ. If (n, B, R) is a triplet in L_θ then n is separated from R given B in the DAG, for if not then there would be a path from a node in R to n which is active given B. But since every link into n is from B the path must lead out of n into some node which was placed after n. Since every node in R was placed before n, this path cannot be directed and must contain a head-to-head node at some node which was placed after n. But this path is deactivated by B since it contains no nodes placed after n, and thus, B would separate n from R in the graph. QED.

The following corollary is needed to assert the tractability of finding an edge-minimal I-map for a graphoid.

Corollary 2: If each tail boundary in L_θ is minimal, the resulting DAG is a minimal I-map of M.

Theorem 2 and its corollaries together imply that d-separation is sound and complete for the extraction of independence information from DAGs with respect to their causal lists when those lists are drawn from graphoids. That is, a conclusion can be read from the graph using d-separation if and only if it follows from application of the graphoid axioms to the causal list. In bayesian networks [8], for example, any independence which can be read from the graph via d-separation is sound with respect to the probability distribution that it represents since the axioms of graphoids are sound for probabilistic dependence. But the axioms of graphoids are not complete for the class of probabilistic dependencies, so corollary 1 is not enough to ensure that d-separation is complete for DAGs built from the more specific probabilistic dependency models. Completeness with respect to probability has been shown in [3].

The last theorem, which is of a theoretical nature, states that it is possible to force any particular independence of a graphoid to be represented in an I-map.

Theorem 3: If M is any graphoid then the set of DAGs generated from all causal lists of M

is a perfect map of M if the criterion for separation is that d-separation must exist in one of the DAGs.

Proof: If there is a separation in one of the DAGs then the corresponding independence must hold in M since theorem 2 states that each of the DAGs is an I-map of M, thus the set is also an I-map. It remains to show that M is an I-map of the set of DAGs. Let $T = (X, Z, Y)$ be any triplet in M and $X = \{x_1, \ldots, x_n\}$. The triplets $T^* = \{(x_i, x_1 \cdots x_{i-1} Z, Y) \mid 1 \leq i \leq n\}$ must also be in M since they are implied by T using the weak union axiom of graphoids. Furthermore T is in the graphoid closure of T^* since the triplets imply T by use of the contraction axiom. Thus any causal list containing the triplets T^* would generate a DAG containing T. Such a list need only have an ordering θ such that the variables of Y and Z are less than those of X which are less than any other variables and that the variables of X are ordered such that $x_i <_\theta x_j$ if and only if $i < j$. The DAG generated by this causal list is in the set of DAGs and therefore the separation holds in the set. QED.

Since there is an effective algorithm for generating an I-map DAG for any graphoid, DAGs are a useful means for representing EMVD relations as well as probabilistic independence relations. Furthermore if the particular dependency model is stated as a causal list then it can be perfectly represented by a DAG.

4. FUNCTIONAL DEPENDENCIES

The ability to represent functional dependencies would be a powerful extension from the point of view of the designer. These dependencies may easily be represented by the introduction of *deterministic nodes* which would correspond to the deterministic variables [13]. Graphs which contain deterministic nodes represent more information than d-separation is able to extract; but a simple extension of d-separation, called D-separation, is both sound and complete with respect to the input list under both probabilistic inference and graphoid inference [4]. D-separation is very similar to d-separation, only differing in that a path is rendered *inactive* by a set of nodes Z under D-separation just in case it would be inactive under d-separation plus the case when a node on the path which is *determined* by Z.

5. CONCLUSIONS

This paper shows that d-separation is sound (i.e. correct) and briefly discusses D-separation which is also sound. They both provide a reliable and efficient method for extracting independence information from DAGs. This information may be used explicitly, for example to help guide a complex reasoning system, or implicitly as in bayesian propagation [8] or the evaluations of Influence Diagrams [12,13]. These criteria also provide a sound theoretical basis for the analysis of the properties of the corresponding graphical representations. For example, the validity of graphical manipulations such as arc reversal and node removal [5,12,13,14] can now be affirmed on solid theoretical foundations.

ACKNOWLEDGMENT

We thank Dan Geiger and Azaria Paz for many valuable discussions, and James Smith for checking the proof of Theorem 2.

REFERENCES

[1] Dawid, A.P., Conditional Independence in Statistical Theory, *J.R. Statist. Soc. B.* (1979) **41(1):** 1-33.

[2] Fagin, R., Multivalued Dependencies and a New Form for Relational Databases, *ACM Transactions on Database Systems* (1977) **2(3):** 262-278.

[3] Geiger D. and Pearl J. On the Logic of Causal Models, this volume.

[4] Geiger, D., Verma, T.S. and Pearl, J., Recognizing Independence in Influence Diagrams with Deterministic Nodes, To appear in *Networks* (1990).

[5] Howard, R.A. and Matheson, J.E., Influence Diagrams. In Howard, R.A. and Matheson, J.E., (eds) *Principles and Applications of Decision Analysis* (Strategic Decision Group, Menlo Park, CA, 1981) **2:** 719-762.

[6] Pearl, J., Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach, *Proc. of the Natl. Conference on AI* (Pittsburgh, 1982) 133-136.

[7] Pearl, J., A Constraint Propagation Approach to Probabilistic Reasoning. In Kanal, L.N. and Lemmer, J. (eds) *Uncertainty in Artificial Intelligence,* (North-Holland, Amsterdam, 1986) 357-369.

[8] Pearl, J., Fusion, Propagation and Structuring in Belief Networks, *Artificial Intelligence* (1986) **29(3):** 241-288.

[9] Pearl, J., *Probabilistic Reasoning in Intelligent Systems* (Morgan-Kaufmann, San Mateo, 1988)

[10] Pearl, J. and Paz, A., GRAPHOIDS: A Graph-based Logic for Reasoning about Relevance Relations. In B. Du Boulay et al. (eds) *Advances in Artificial Intelligence-II* (North-Holland, Amsterdam, 1987).

[11] Pearl, J. and Verma, T.S., The Logic of Representing Dependencies by Directed Graphs, *Proc. of the 6th Natl. Conference on AI* (Seattle, 1987) **1:** 374-379.

[12] Shachter, R., Intelligent Probabilistic Inference. In Kanal, L.N. and Lemmer, J. (eds) *Uncertainty in Artificial Intelligence,* (North-Holland, Amsterdam, 1986) 371-382.

[13] Shachter, R., Probabilistic Inference and Influence Diagrams, *Operations Research* (1988) **36:** 589-604.

[14] Verma, T.S., Some Mathematical Properties of Dependency Models, *Technical Report R-102* (UCLA Cognitive Systems Laboratory, Los Angeles, 1987)