CAUSAL NETWORKS: SEMANTICS AND EXPRESSIVENESS

Thomas Verma

July 1987
CSD-870032

TECHNICAL REPORT
R-65-I
June 1987

Causal Networks: Semantics and Expressiveness

by

Thomas Verma

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

ABSTRACT

Dependency knowledge of the form "x is independent of y once z is known" can often be stored efficiently in graphical structures. Both undirected graphs, and DAGs (directed acyclic graphs) have been studied for this purpose. It is shown that DAGs constructed from stratified protocols do in fact perfectly represent the underlying dependency model whenever possible. Further, if the underlying model is a *semi-graphoid* then the DAG generated by any protocol is an *I-map* of the model (i.e. produces sound assertions of independence) and the set of all DAGs generated is a perfect map. Finally the possibility of using hybrid graphs with both directed and undirected links is introduced and shown to be more expressive than the union of the previous two representations.

Science track

Major topic: Knowledge Representation

Subtopics: Data Dependencies, Logic of Relevance, Network Representations

* This work was supported in part by the National Science Foundation Grants #DCR 85-01234 and #IRI 86-10155

Introduction

Dependency knowledge is useful in several areas of research, for example in database design it is useful to reason about embedded-multivalued-dependence (EMVD) of attributes [Fagin, 1977] and in expert systems it is useful to reason about probabilistic independence of variables [Pearl, 1986a]. These examples represent two formalizations of the intuitive relation "knowing Z renders X and Y independent" which shall be denoted as $I(X, Z, Y)$. This relation would naturally have different properties in different applications, but it is interesting to note that most sensible definitions of this relation share 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, attributes). It is known that every EMVD relation obeys the four properties listed above, as well as many other properties. (The notation $I(X, Z, Y)$ is equivalent to the standard EMVD notation $Z \twoheadrightarrow X \mid Y$). Probabilistic dependencies also obey these four properties, and it has been conjectured that they are, in fact, complete [Pearl and Paz, 1986], namely, that any other property of probabilistic independence is a logical consequence of the four. Three place relations which obey these four properties are called *semi-graphoids* [Dawid, 1979].

It is worth noting that for probability distributions containing strictly positive probabilities, the independence relation has a fifth independent property:

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

This property along with the four of semi-graphoids define the class of *graphoids* (also conjectured to be complete for non-extreme probabilities [Pearl and Paz, 1986]).

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 result in exponential space since the relation I ranges over subsets of objects. The use of graphs as a representation of dependency models is appealing in two ways; first the graph has an intuitive conceptual meaning, and second, it is an efficient representation in terms of time and space [Pearl and Verma, 1987].

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. Under this meaning, 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 [Pearl and Paz, 1986]:

$$\text{symmetry} \qquad I(X, Z, Y) \Leftrightarrow I(Y, Z, X) \qquad (3.a)$$

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* $I(Y, Z, \gamma)\ \forall \gamma \notin X \cup Y \cup Z$	(3.e)

It is not always necessary to have an exact graphical 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 of a dependency model M is an *I-map* if every independence represented in R implies a valid independence in M. Thus, R may not contain all independencies of M, but the ones it does contain are correct. There is an algorithm which finds the most representative *I-map* for any graphoid [Pearl and Paz, 1986]. Since probabilistic independence over positive probabilities constitutes a graphoid, there is always a unique edge-minimal undirected graph which is an *I-map* of any probabilistic distribution P. This is not the case for EMVD relations; there is no unique edge-minimal *I-map* for a given database.

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 [Pearl, 1986b]. Thus a set of nodes Z is defined to *d-separate* two other sets X and Y if and only if every *adjacency path* from a node in X to a node in Y is rendered *inactive* by Z. An *adjacency path* is one which follows arcs ignoring their directionality; one is rendered *inactive* by a set of nodes Z if and only if either there is a *head-to-head* node along the path which is not in Z and none of its *descendents* are in Z or some node along the path is not *head-to-head* but is in Z. A node along the path is *head-to-head* if the node before it and after it along the path both point to it in the graph. One node is a *descendent* of another if there is a directed path from the latter to the former.

A complete set of axioms which define the class of dependency models representable by a DAG has not yet been determined [Geiger, 1987], nor has an algorithm been found which finds an optimal *I-map*. But there is an algorithm which produces a perfect map of a dependency model if such exists. Actually the algorithm takes a *stratified protocol* of a dependency model and produces a perfect map of its semi-graphoid closure. A *stratified protocol* of a dependency model contains two things: an ordering of the variables, and a function that assigns a *tail boundary* to each variable x. A *tail boundary* of a variable x is any set of *lesser* variables (with respect to the ordering) rendering x independent of all other *lesser* variables. A unique DAG can be generated from each *stratified protocol* by associating the set of direct parents of any node x in the DAG with the *tail boundary* of the variable x in the protocol. An equivalent specification of a *stratified protocol* is an ordered list of triplets of the form $I(n, B, R)$, one triplet for each variable in the model, where the set B is the *tail boundary* of the variable n and R is a set containing all other *lesser* variables. For a particular dependency model over n variables there are $n!$ orderings, and for each ordering there can be up to $\prod_{k=1}^{n} 2^{k-1} = 2^{n(n-1)/2}$ 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 $n!\,2^{n(n-1)/2}$ stratified protocols. But if the dependency model posses a perfect map in DAGs, then one of the proto-

cols is guaranteed to generate it.

Theorem 1: If M is a dependency model which can be perfectly represented by some DAG D, then there is a stratified protocol 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 n in D, the set of its parents $P(n)$ constitutes a *tail boundary* with respect to the ordering θ, thus the pair $L_\theta = (\theta, P(n))$ is a stratified protocol of M, and this is the very protocol which will generate D. QED.

The next theorem shows that stratified protocols can be used to generate *I-maps* of any semi-graphoid, not necessarily those possessing perfect maps in DAGs.

Theorem 2: If M is a semi-graphoid, and L_θ is any stratified protocol of M, then the DAG generated by L_θ is an *I-map* of M.

Proof: Induct on the number of variables in the semi-graphoid. For semi-graphoids of one variable it is obvious that the DAG generated is an *I-map*. Suppose for semi-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 semi-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 stratified protocol 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.

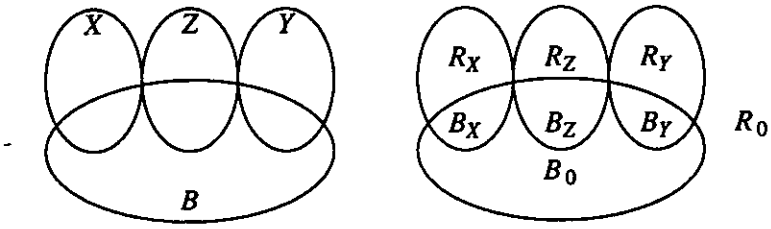

Figure 1

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 semi-graphoid it follows (from (1.b) and (1.c)) that $(n, XB_0Z, Y) \in M$.

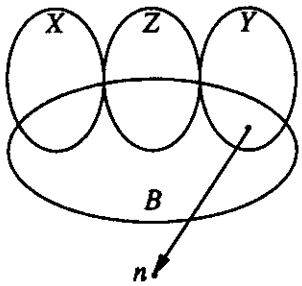

Figure 2a

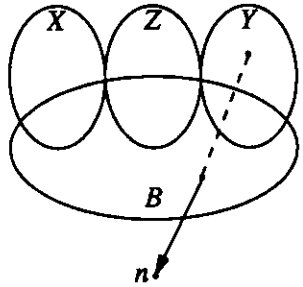

Figure 2b

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 semi-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 $T = (X, Zn, Y)$ with the same constraints on X, Y and Z as in case-1. Also let B, R, etc. be defined as in case-2, thus $(n, B_X B_Y B_Z B_0, R_X R_Y R_Z R_0) \in M$ since it is the last triplet in L_θ.

In this case, either B_y is empty and B_0 is separated from Y given Z in G, or B_X is empty and B_0 is separated from X given Z in G since if neither were the case, then there would be a path from a node in Y which would be active given Z and would end *pointing* at n, and there would be a similar path from a node in X to n. But this means that there would be a path from a node in X to a node in Y which would be active given Z and n since the path is head-to-head at n (see figure 3a and 3b). But there can be no such path

since by assumption $(X, Zn, Y) \in M_G$. Without loss of generality, assume that $B_Y = \varnothing$ and $(B_0, Z, Y) \in M_G$.

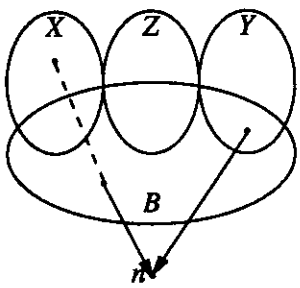

Figure 3a

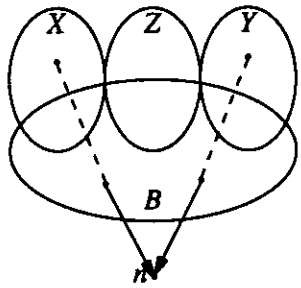

Figure 3b

X and Y must be separated in G given only Z for if they were not then there would be a path between them which would be active given Z. Since they are separated given Z and n, this path would have to be deactivated by n, but since there are only arrows pointing at n it can only activate paths by being instantiated (see figure 3b). Thus, there can be no such path and X and Y must be separated given Z in G. Since Y is separated from both X and B_0 given Z in G it follows that $(XB_0, Z, Y) \in M_G$ and by the argument of case-1 above $(XB_0, Z, Y) \in M$. Since B_Y is empty it follows, as in case-2, that $(n, XB_0Z, Y) \in M$. Further, M is a semi-graphoid so, applying (1.d) to $(XB_0, Z, Y) \in M$ and $(n, XB_0Z, Y) \in M$ yields $(nXB_0, Z, Y) \in M$ and using (1.b) and (1.c) it follows that $T = (X, Zn, Y) \in M$.

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.

Corollary: If L_θ is any stratified protocol of some dependency model M, the DAG generated from L_θ is a perfect map of the semi-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 triplets of L_θ using the four axioms in (1).

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 semi-graphoid, the DAG closure of L_θ contains the semi-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.

Theorem 3: If M is any semi-graphoid then the set of DAGs generated from all *stratified protocols* 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 $T^* = \{(x_i, x_1 \cdots x_{i-1} Z, Y) \mid 1 \le i \le n\}$ must also be in M since they are implied by T using the weak union axiom of semi-graphoids. Furthermore T is in the semi-graphoid closure of T^* since the triplets imply T by use of the contraction axiom. Thus any protocol containing the triplets T^* would generate a DAG containing T. Such a protocol 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 protocol is in the set of DAGs and therefore the separation holds in the set. QED.

This set is not the smallest set of DAGs which is a perfect map, in fact it may contain many redundant DAGs. A *tail boundary* B of a variable n is minimal if there is no other *tail boundary* B' which is a proper subset of it. A *stratified protocol* is minimal if it contains only minimal boundaries. The following corollary states that only the DAGs built from minimal protocols need be consulted for a perfect map.

Corollary: If M is any semi-graphoid then the set of DAGs generated from all minimal *stratified protocols* of M is a perfect map of M.

Proof: Theorem 3 states that the set of DAGs generated from all protocols is a perfect map, it is enough to show that any DAG built from a non-minimal protocol is subsumed by a DAG built from a minimal protocol. Suppose L_θ is a non-minimal protocol. Let L'_θ be any minimal protocol with the ordering θ such that every boundary in L'_θ is a subset of the corresponding boundary in L_θ. L'_θ must exist and the DAG it generates (G') is a subset of the DAG generated from L_θ (G) since the parent sets of G' are subsets of the corresponding parent sets of G. Thus the DAG G' which is built from a minimal protocol L'_θ does subsume the DAG G, built from L_θ. QED.

Even though this reduces the number of DAGs necessary for a perfect map, it is not an effective perfect map since there is at least one minimal protocol for each ordering, thus there will be at least $n!$ minimal protocols.

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

Power and Limitations

It is obvious that both undirected graphs and DAGs have limitations, otherwise there would be no need to settle for *I-maps*. Undirected graphs can only perfectly represent dependency models which are strongly transitive and have the a strong union property (see (3)), whereas DAGs can only represent weakly transitive and chordal dependency models [Pearl, 1986b]. Furthermore, both can only represent

dependency models which have the intersection and set composition properties. But any graphical representation would suffer from these last two restrictions unless the semantics were drasticly altered. It is interesting to note that the set of dependency models representable by undirected graphs overlaps that of dependency models representable by DAGs, each having models not representable by the other. The two intersect in the class of chordal graphs [Pearl and Verma, 1987].

These limitations also affect the expressive power of *I-maps*; the more limited the representation is, the fewer independencies a particular *I-map* can display. In fact if a relation is extremely incompatible with the particular representation even the best *I-maps* say nothing. Take for example the situation where a bell (controlled by an oracle) will ring every time two coins are tossed and land the same, i.e. both heads or both tails. The variables in this situation are the outcome of each coin, and the outcome of the bell. Any two of the variables are pairwise independent, (e.g. knowing the outcome of the first coin tells nothing about the outcome of the second), but once any one is known the other two become dependent. The only undirected graph *I-map* for this situation is a complete graph which says nothing. The three DAGs representing this situation have the structure $\bullet\rightarrow\bullet\leftarrow\bullet$, each asserting one independence triplet.

Hybrid Graphs

Consider a graph which contains both directed and undirected links, the directed links would denote a direct causal relationship, and undirected links would represent symmetric correlations. The criterion for separation in these hybrid acyclic graphs is almost identical to that for DAGs -- two sets are *h-separated* given a third if and only if every *adjacency* path between them is rendered *inactive* by the third. An *adjacency* path, which may also contain undirected links, is rendered *inactive* by a set Z if and only if a node along the path is in Z and it is not *head-to-head*, or if there is a *head-to-head* node on the path and neither it nor any of its *h-descendents* are in the set. Here, however, a node is an *h-descendent* of another if there is an *adjacency path* between the ancestor to the descendent in which any directed arcs point in the direction of the descendent. The reasoning behind this is that observing the outcome of an event which is correlated with the common effect of two unrelated causes should serve to correlate them just as the outcome of the effect itself would.

This definition is a extension of both the undirected and the directed acyclic graph definitions. Trivialy it is as powerful as their union because any dependency model which is representable by either a undirected graph or a DAG is also representable by a hybrid graph. Furthermore, there are dependency models representable by hybrid acyclic graphs which are not representable by either undirected graphs or DAGs individually. In fact the dependency model can be both non-chordal and non-transitive (see figure 4), two properties that no DAG nor any undirected graph can display individually.

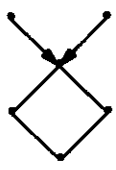

Figure 4

Conclusion

Since hybrid graphs are a good generalization of undirected and directed acyclic graphs, exceeding the expressive power of both combined, it would be useful to investigate them further and find efficient or even theoretical algorithms to generate them. Their additional power is useful in two ways, first it allows a larger class of dependency models to be perfectly representable by graphs and, second, it allows the construction of *I-maps* that display more facts about the partially represented model.

REFERENCES

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

R. Fagin, "Multivalued Dependencies and a New Form for Relational Databases," *ACM Transactions on Database Systems*, 2, 3,; September 1977, pp. 262-278.

D. Geiger, "The Non-axiomatizability of Dependencies in Directed Acyclic Graphs," *Technical Report R-83*, Cognitive Systems Laboratory, UCLA. 1987.

J. Pearl, "Fusion, Propagation and Structuring Belief Networks," *Artificial Intelligence*, Vol. 29, No 3, September 1986, pp. 241-288

J. Pearl, "Bayes and Markov Networks: a Comparison of Two Graphical Representations of Probabilistic Knowledge," UCLA Computer Science Department *Technical Report 860024* (R-46), October 1986.

J. Pearl & A. Paz, "GRAPHOIDS: a Graph-based Logic for Reasoning about Relevance Relations," *Proceedings, ECAI-86*, Brighton, U.K., June 1986; also, UCLA Computer Science Department *Technical Report 850038 (R-53)*.

J. Pearl & TS Verma, "The Logic of Representing Dependencies by Directed Graphs," *Proceedings*, AAAI-87, Seattle, WA, July 1987