The Logic of Representing Dependencies by Directed Graphs*

Judea Pearl & Thomas Verma

Cognitive Systems Laboratory
UCLA Computer Science Department, Los Angeles, CA 90024-1600

Abstract

Data-dependencies of the type "x can tell us more about y given that we already know z" can be represented in various formalisms: Probabilistic Dependencies, Embedded-Multi-Valued Dependencies, Undirected Graphs and Directed-Acyclic Graphs (DAGs). This paper provides an axiomatic basis, called a *semi-graphoid*, which captures the structure common to all four types of dependencies and explores the expressive power of DAGs in representing various types of data dependencies. It is shown that DAGs can represent a richer set of dependencies than undirected graphs, that DAGs completely represent the closure of their specification bases, and that they offer an effective computational device for testing membership in that closure as well as inferring new dependencies from given inputs. These properties might explain the prevailing use of DAGs in causal reasoning and semantic nets.

I. Introduction

A. Why Dependencies?

The notion of *relevance* or *informational dependency* is basic to human reasoning. People tend to judge the 3-place relationships of mediated dependency (i.e., x influences y via z) with clarity, conviction and consistency. For example, knowing the departure time of the last bus is considered relevant for assessing how long we are about to wait for the next bus. Yet, once we learn the current whereabouts of the next bus, the former no longer provides useful information. These common-sensical judgments are issued qualitatively and reliably and are robust to the uncertainties which accompany the assessed events. Consequently, if one aspires to construct common-sensical reasoning systems, it is important that the language used for representing knowledge should facilitate a quick detection of mediated dependencies by a few primitive operations on the salient features of the representation scheme.

Making effective use of information about dependencies is a computational necessity, essential in any reasoning. If we have acquired a body of knowledge z and now wish to assess the truth of proposition x, it is important to know whether it would be worthwhile to consult another proposition y, which is not in z. In the absence of such information, an inference engine would spend precious time on derivations bearing no relevance to the task at hand. A similar necessity exists in

truth maintenance systems. If we face a new piece of evidence, contradicting our previously held assumptions, we must retract some of these assumptions and, again, the need arises of quickly identifying those that are *relevant* to the contradiction discovered. But how would relevance information be encoded in a symbolic system?

Explicit encoding is clearly impractical because the number of (x, y, z) combinations needed for reasoning tasks is astronomical. Relevance or dependencies are relationships which change dynamically as a function of the information available at any given time. Acquiring new facts may destroy existing dependencies as well as create new ones. For example, learning a child's age destroys the dependency between the size of his shoes and his reading ability, while learning that a patient suffers from a given symptom creates new dependencies between the diseases that could account for the symptom. What logic would facilitate these two modes of reasoning?

B. Why Logic?

In probability theory, the notion of informational relevance is given precise quantitative underpinning using the device of *conditional independence,* which successfully captures our intuition about how dependencies should change with learning new facts. A variable x is said to be independent of y, given the information z, if

$$P(x, y \mid z) = P(x \mid z)P(y \mid z). \qquad (1)$$

Clearly, x and y could be marginally dependent (i.e., dependent, when z is unknown) and become independent given z, and, conversely, x and y could be marginally independent and become dependent only upon learning the value of z. These dynamics are also captured by the qualitative notion of Embedded Multivalued Dependencies (EMVD) in relational databases. Thus, in principle, probability and database theories could provide the machinery for identifying which propositions are relevant to each other with any given state of knowledge.

Yet, it is flatly unreasonable to expect people or machines to resort to numerical equalities or relational tables in order to extract relevance information. Human behavior suggests that such information is inferred qualitatively from the organizational structure of human memory. Accordingly, it would be interesting to explore how assertions about

relevance can be inferred qualitatively and whether assertions similar to those of probabilistic or database dependencies can be derived *logically* without references to numbers or tables. Preliminary work related to probabilistic dependencies has been reported in [Pearl and Paz, 1986] and is extended in this paper to the qualitative notion of EMVD.

C. Why Graphs?

Having a logic of dependency might be useful for testing whether a set of dependencies asserted by an expert is self-consistent and might also allow us to infer new dependencies from a given initial set of such relationships. However, such logic would not, in itself, guarantee that the inferences required would be computationally tractable or that any sequence of inferences would be psychologically meaningful, i.e., correlated with familiar mental steps taken by humans. To facilitate this latter feature, we must also make sure that most derivational steps in that logic correspond to simple local operations on structures depicting common-sensical associations. We call such structures *dependency graphs*.

The nodes in these graphs represent propositional variables, and the arcs represent local dependencies among conceptually-related propositions. Graph representations are perfectly suited for meeting the requirements of explicitness, saliency and stability, i.e., the links in the graph permit us to qualitatievely encode dependence relationships, and the graph topology displays these relationships explicitly and preserves them, in fact, under any assignment of numerical parameters.

It is not surprising, therefore, that graphs constitute the most common metaphor for describing conceptual dependencies. Models for human memory are often portrayed in terms of associational graphs (e.g., semantic networks [Woods, 1975], constraint networks [Montanari, 1974], inference nets [Duda, Hart and Nilsson, 1976] conceptual dependencies [Schank 1972]) and conceptual graphs [Sowa, 1983]). Graph-related concepts are so entrenched in our language (e.g. "threads of thoughts," "lines of reasoning," "connected ideas," "far-fetched arguments" etc.) that one wonders whether people can, in fact, reason any other way except by tracing links and arrows and paths in some mental representation of concepts and relations. Therefore, a natural question to ask is whether the intuitive notion of informational relevancy or the formal notions of probabilistic and database dependencies can be captured by graphical representation, in the sense that all dependencies and independencies in a given model would be deducible from the topological properties of some graph.

Despite the prevailing use of graphs as metaphors for communicating and reasoning about dependencies, the task of capturing dependencies by graphs is not at all trivial. When we deal with a phenomenon where the notion of neighborhood or connectedness is explicit (e.g., family relations, electronic circuits, communication networks, etc.), we have no problem configuring a graph which represents the main features of the phenomenon. However, in modeling conceptual relations such as causation, association and relevance, it is often hard to

distinguish direct neighbors from indirect neighbors; so, the task of constructing a graph representation then becomes more delicate. The notion of conditional independence in probability theory provides a perfect example of such a task. For a given probability distribution P and any three variables x, y, z, while it is fairly easy to verify whether knowing z renders x independent of y, P does not dictate which variables should be regarded as direct neighbors. In other words, we are given the means to test whether any given element z *intervenes* in a relation between elements x and y, but it remains up to us to configure a graph that encodes these interventions. We shall see that some useful properties of dependencies and relevancies cannot be represented graphically and the challenge remains to devise graphical schemes that minimize such deficiencies.

D. Why Directed Graphs?

Ideally, we would like to represent dependency between elements by a path connecting their corresponding nodes in some graph G. Similarly, if the dependency between elements x and y is not direct and is mediated by a third element, z, we would like to display z as a node that intercepts the connection between x and y, i.e., z is a cutset separating x from y. This correspondence between conditional dependencies and cutset separation in undirected graphs forms the basis of the theory of Markov fields [Lauritzen, 1982], and has been given an axiomatic characterization in [Pearl and Paz, 1986].

The main weakness of undirected graphs stems from their inability to represent nontransitive dependencies; two independent variables will end up being connected if there exists some other variable that depends on both. As a result, many useful independencies remain unrepresented in the graph. To overcome this deficiency, one can employ *directed* graphs and use the arrow directionality to distinguish between dependencies in various contexts. For instance, if the sound of a bell is functionally determined by the outcomes of two coins, we will use the network *coin* 1 → *bell* ← *coin* 2, without connecting *coin 1* to *coin 2*. This pattern of converging arrows is interpreted as stating that the outcomes of the two coins are normally independent but may become dependent upon knowing the outcome of the bell (or any external evidence bearing on that outcome). This facility of directed graphs forms the basis of causal networks which have a long tradition in the social sciences [Kenny, 1979], and have also been adopted for evidential reasoning tasks [Pearl, 1986].

This paper treats directed graphs as a language of expressing dependencies. Section II presents formal definitions for two models of data dependencies (Probabilistic and EMVD) and two models of graphical dependencies (undirected and directed). An axiomatic definition is then provided for a relational structure called *semi graphoid* which covers all four models, thus formalizing the general notion of mediated dependence. Section III compares the expressive power of directed graphs to that of undirected graphs and shows the superiority of the former. Section IV, explores the power of directed graphs to cover data dependencies of the type produced by probabilistic or logical models. The main contribu-

tion of the paper lies in showing that directed acyclic graphs (DAGs) are powerful tools for encoding and inferring data dependencies of both types, identifying the scource of that power, and highlighting its limitations.

II. Dependency Models, Maps and Graphoids

Definition: A *Dependency Model M* over a set of objects U is any subset of triplets (X, Z, Y) where X, Y and Z are three disjoint subsets of U. The triplets in M represent independencies, that is, $(X, Z, Y) \in M$ asserts that X and Y interact only via Z, or, "X is independent of Y given Z". This statement is also written as $I(X, Z, Y)$ with an optional subscript to clarify the type of the dependency when necessary.

Definition: A *Probabilistic Dependency* model (PD) M_P is defined in terms of a probability distribution P over some set of variables U, i.e. a function mapping any instantiation of the variables in U to a non-negative real number such that the sum over the range of P is unity. If X, Y and Z are three subsets of U and x, y and z any instantiation of the variables in these subsets, then by definition $I(X, Z, Y)_P$ iff

$$P(x \mid zy) = P(x \mid z) \qquad (2)$$

This definition is equivalent to that given in (1) and conveys the idea that, once Z is fixed, knowing Y can no longer influence the probability of X. [Dawid, 1979].

Definition: A dependency model M is said to be *in* PD, $M \in$ PD, if there exists a probability distribution P such that the definition above (Eq.(2)) holds for every triplet (X, Z, Y) in M. Thus, PD (and, similarly, PD⁻, UGD, DAGD, and SG defined below) represents a *class* of dependency models, all sharing a common criterion for selecting triplets in M.

Definition: A *Non-Extreme Probabilistic Dependency* model (PD⁻) is any model M_P in PD where the range of P is restricted to the positive real numbers, (i.e., excluding 0's and 1's).

Definition [Fagin, 1977]: An *Embedded Multivalued Dependency* model (EMVD) M_R is defined in terms of a database R over a set of attributes U, i.e. a set of tuples of values of the attributes. The notation $<a_1 a_2 \cdots a_n>$ is conventionally used to denote that the tuple is in the relation R. If X, Y and Z are three disjoint subsets of U and x_1, x_2, y_1, y_2, z any instantiations of the corresponding attributes in X, Y and Z, then by definition $I(X, Z, Y)_R$ iff

$$<x_1 y_1 z > \& <x_2 y_2 z > \Rightarrow <x_1 y_2 z > \qquad (3)$$

In other words, the existence of the subtuples $<x_1 y_1 z >$ and $<x_2 y_2 z >$ guarantees the existence of $<x_1 y_2 z >$. EMVD is a powerful class of dependencies used in databases and it conveys the idea that, once Z is fixed, knowing Y cannot further restrict the range of values permitted for X. This definition was also used by Shenoy and Shafer [1986] to devise a "qualitative" extension of probabilistic dependencies.

Definition: An *Undirected Graph Dependency* model (UGD) M_G is defined in terms of an undirected graph G. If X, Y and Z are three disjoint subsets of nodes in G then by definition $I(X, Z, Y)_G$ iff every path between nodes in X and Y contains at least one node in Z. In other words, Z is a cutset separating X from Y. A complete axiomatization of UGD is given in [Pearl and Paz, 1986].

Definition: A *Directed Acyclic Graph Dependency* model (DAGD) M_G is defined in terms of a directed acyclic graph (DAG) G. If X, Y and Z are three disjoint subsets of nodes in G, then by definition $I(X, Z, Y)_G$ iff there is no bi-directed path from a node in X to a node in Y along which every node with converging arrows either is or has a descendent in Z and every other node is outside Z.

The latter condition corresponds to ordinary cutset separation in undirected graphs while the former conveys the idea that the inputs of any causal mechanism become dependent once the output is known. This criterion was called *d-separation* in [Pearl, 1986]. In Figure 1, for example, $X = \{2\}$ and $Y = \{3\}$ are d-separated by $Z = \{1\}$ (i.e. $(2, 1, 3) \in M_G$) because knowing the common cause 1 renders its two possible consequences, 2 and 3, independent. However X and Y are not d-separated by $Z' = \{1, 5\}$ because learning the value of the consequence 5, renders its causes 2 and 3 dependent, like opening a pathway along the converging arrows at 4.
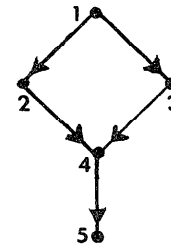

Figure 1. A DAG displaying d-separation: $(2, 1, 3) \in M_G$ while $(2, \{1, 5\}, 3) \notin M_G$

Definition: An *I-map* of a dependency model M is any model M' such that $M' \subseteq M$. For example, the undirected graph $X - Z - Y$ is an I-map of the DAG $X \rightarrow Z \leftarrow Y$.

Definition: A *D-map* of a dependency model M is any model M' such that $M' \supseteq M$. For example, if a relation R contains all tuples having non-zero probability in P then M_R is a D-map of M_P.

Definition: A *Perfect-map* of a dependency model M is any model M' such that $M' = M$. For example, the undirected graph $X - Z - Y$ is a perfect map of the DAG $X \rightarrow Z \rightarrow Y$.

We will be primarily interested in mapping data dependencies into graphical structures, where the task of testing connectedness is easier than that of testing membership in the original model M. A D-map guarantees that vertices found to be connected are, indeed, dependent; however, it may occasionally display dependent variables as separated vertices. An I-map works that opposite way: it guarantees that vertices

found to be separated always correspond to genuinely independent variables but does not guarantee that all those shown to be connected are, in fact, dependent. Empty graphs are trivial D-maps, while complete graphs are trivial I-maps.

Definition: A *semi-graphoid* (SG) is any dependency model M which is closed under the following properties:

Symmetry: $(X,Z,Y) \in M \iff (Y,Z,X) \in M$

Decomposition $(X,Z,YW) \in M \Rightarrow (X,Z,Y) \in M$

Weak Union $(X,Z,YW) \in M \Rightarrow (X,ZW,Y) \in M$

Contraction $(X,ZY,W) \& (X,Z,Y) \in M \Rightarrow (X,Z,YW) \in M$ (4)

It is straight forward to show that all the specialized classes of dependency models presented thus far are semi-graphoids, and in view of this generality, these four properties are selected to represent the general notion of mediated dependence between items of information.

With the exception of UGD, none of the specialized dependency classes possesses complete parsimonious axiomatization similar to that of semi-graphoids. EMVD is known to be non-axiomatizable by a bounded set of Horn clauses [Parker, 1980], and a similar result has recently been reported for DAGD [Geiger, 1987]. PD is conjectured to be equivalent to SG (i.e., $M \in PD \iff M \in SG$) but no proof (nor disproof) is in sight.

Definition: A *graphoid* is any semi-graphoid M which is also closed under the following property:

Intersection:

$(X,ZY,W) \& (X,ZW,Y) \in M \Rightarrow (X,Z,YW) \in M$ (5)

It is straight forward to show that classes PD⁻, UGD, and DAGD are all graphoids. Only EMVDs and pure PDs do not comply to this axiom.

The most important properties of graphoids [Pearl and Paz, 1986] are that they possess unique edge-minimal I-maps in UG, and permit the construction of graphical I-maps from local dependencies. By connecting each variable x to any subset of variables which renders x conditionally independent of all other variables in U, we obtain a graph that is an I-map of U. Such local construction is not guaranteed for semi-graphoids. The reason this paper focuses on semi-graphoids is to include dependency models representing logical, functional and definitional constraints; such constraints are excluded from PD⁻. In Section IV, we will show that the use of DAG's provides a local construction of I-maps for every semi-graphoid. The relationships between the six classes of dependency models are shown in the hierarchy of Figure 2, where arrows stand for set inclusions.
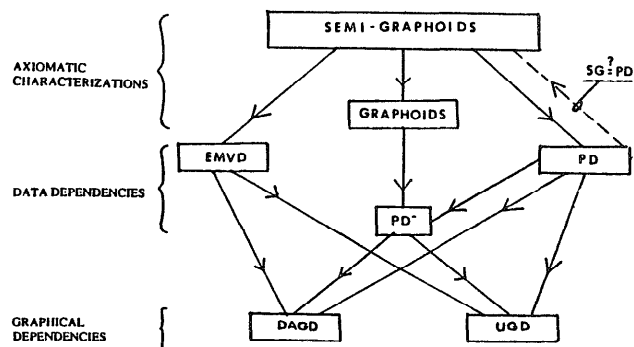

Figure 2. Hierarchy among six classes of dependency models.

Definition: Let M be a dependency model from some class M of dependency models. A subset $B \subseteq M$ of triplets is a M-*basis* of M iff every model $M' \in M$ which contains B also contains M. Thus, a basis provides a complete encoding of the information contained in M; knowing B and M enables us, in principle, to decide what triplets belong to M.

One of the main advantages of graphical representations is that they posses extremely parsimonious bases and extremely efficient procedures for testing membership in the closures of these bases. For example, to encode all dependencies inferable from a given undirected graph $G = (V, E)$ we need only specify the set of neighbors $N(x)$ for each node x in G, and this corresponds to specifying a neighborhood basis:

$$B_N = \{(x, N(x), U - x - N(x)), \forall x \in V\}$$ (6)

Testing membership of an arbitrary triplet (X, Z, Y) in the closure of B_N simply amounts to testing whether Z is a cutset of G separating the nodes in X from those in Y.

DAGs also possess efficient bases; to encode all dependencies inferable from a given DAG G, we need only specify the parents $PA(x)$ for each node $x \in G$. To encode those in the form of a basis we arrange the nodes in any total order $x_1, \cdots, x_n$ consistent with the arrows of G and construct the stratified set of triplets:

$$B = \{(x_i, PA(x_i), \{x_1, \ldots, x_{i-1}\} - PA(x_i)) \mid i = 1, \ldots, n\},$$ (7)

stating that $PA(x_i)$ d-separates x_i from its other predecessors. B is a DAG-basis of G since the closure of B coincides with the independencies displayed by G.

III. How Expressive are DAGs?

One would normally expect that the introduction of directionality into the language of graphs would render them more expressive, capable of portraying a more refined set of dependencies, e.g., non-transitive. Thus, it is natural to ask:

1. Are all dependencies representable by undirected graphs also representable by a DAG?

2. How well can DAGs represent the type of data dependencies induced by probabilistic or logical models?

The second question will be treated in Section IV while the answer to the first question is, clearly, negative. For instance, the dependency structure of the diamond-shaped graph of Fig.3(a) asserts the two independencies: $I(A,BC,D)$ and $I(B,AD,C)$. No DAG can express these two relationships simultaneously and exclusively. If we direct the arrows from A to D, we get $I(A,BC,D)$ but not $I(B,AD,C)$; if we direct the arrows from B to C, we get the latter but not the former. This limitation will always be encountered in non-chordal graphs, i.e., graphs containing a chordless cycle of length ≥ 4 [Tarjan & Yannakakis, 1984]; no matter how we direct the arrows, there will always be a pair of non-adjacent parents sharing a common child, a configuration which yields independence in undirected graphs but dependence in DAGs. This problem does not exists in chordal graphs and, consequently, we have

Theorem 1: UGD and DAGD intersect in a class of dependency models representable by chordal graphs.

Non-chordal graphs represent the one class of dependencies where undirected graphs exhibit expressiveness superior to that of DAGs graphs. However, this superiority can be eliminated by the introduction of auxiliary variables. Consider the diamond-shaped graph of Figure 3(a). Introducing an auxiliary variable E in the manner shown in Figure 3(b) creates a DAG model on five variables which also asserts $I(C,B,D)$.
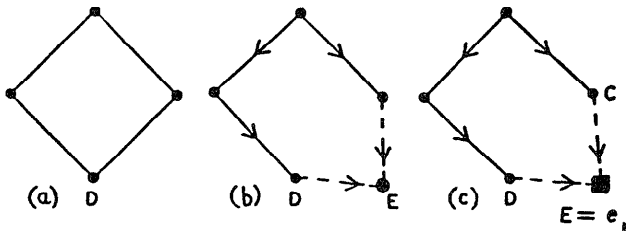

Figure 3. Expressing dependencies of undirected graph (a) by a DAG (c) using auxiliary nodes.

If we "clamp" the auxiliary variable E at some fixed value $E = e_1$, as in Figure 3(c), the dependency structure projected on A,B,C,D is identical to the original structure of Figure 3(a), i.e., $I(A,BC,D)$ and $I(B,AD,C)$.

In general, since every arc $C-D$ in an undirected graph is equivalent to the bi-directed path $C\rightarrow E\leftarrow D$ (with E "clamped"), we have:

Theorem 2: Every dependency model expressible by an undirected graph is also expressible by a DAG, with some auxiliary nodes.

IV. How Faithful are DAGs?

Suppose someone (e.g., an expert) provides us with a list L of positive and negative triplets, representing a set of independencies and dependencies in some (undisclosed) dependency model M, of a known class. Several questions arise:

1. How can we test whether L is consistent and/or non redundant?

2. How can we deduce all the implications of L, or, at least test whether a given triplet is logically implied by L?

3. What additional triplets are required to make the model completely specified?

These questions are extremely difficult to answer if M does not possess a convenient basis or if L does not coincide with that basis. Even in a neatly axiomatized system such as semi-graphoids the answers to these questions involve intractable proof procedures.

Graph representations can be harnessed to alleviate these difficulties; we construct a graph model that entails L and draw inferences from G instead of L. The quality of inferences will depend, of course, on how faithfully G captures the closure of L. The following results (see [Verma, 1987] for proofs) uncover the unique powers of DAGs in performing this task.

Let $U_{\theta(n)}$ represent the set of elements smaller than n under some total ordering θ on the elements of U, i.e., $\{u \in U \mid \theta(u) < \theta(n)\}$.

Definition: A *stratified protocol* L_θ of a dependency model M is any set of pairs

$$L_\theta = \{(x, S_x) \mid \forall\ x \in U, S_x \subseteq U_{\theta(x)}\} \qquad (8)$$

such that

$$(x, S_x) \in L_\theta \Rightarrow (x, S_x, U_{\theta(x)} - S_x) \in M \qquad (9)$$

Intuitively, L_θ lists, for each $x \in U$, a set of predecessors S_x of x which renders x conditionally independent of all its other predecessors (in the order θ). In causal modeling, L_θ specifies the set of direct causes of event x. For example, the causal model of Figure 1 is specified by the protocol:

$$L_\theta = \left\{ (1, \{0\}), (2, \{1\}), (3, \{1\}), (4, \{2, 3\}), (5, \{4\}) \right\}$$

Stratified protocols were used in [Pearl 1986] to construct DAG representations (called Bayesian Networks) of probabilistic dependencies by connecting the elements in S_x as direct parents of x. The following results justify this construction and generalize it to any semi-graphoid, including, in particular, the qualitative dependencies of EMVD.

The first result states that the DAG constructed in this fashion can faithfully be used to infer dependency information; any independence inferred from that DAG must be true in M and, furthermore, every independence which is implied by the protocol will be displayed in the DAG.

Theorem 3: If M is any semi-graphoid then the DAG generated from any stratified protocol L_θ of M is an I-map of M.

Corrollary: 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.

Another interesting corollary of Theorem 3 is a generalization of the celebrated *Markov-chain* property. It states (informally) that if in a sequence of variables $X_1, X_2, \cdots, X_i \cdots$ each X_i "shields" its successor X_{i+1} from the influence of its predecessors, then each X_i is "shielded" from all other variables by its two nearest neighbors, X_{i-1} and X_{i+1}. (The converse holds only in graphoids). This property has been used extensively in probability theory and Theorem 3 permits its application to qualitative dependencies as well.

Note that, since the topology of the DAG depends only on the set of child-parents pairs contained in the protocol, the order θ used in generating L_θ need not be known; Theorem 3 holds for any generating order, and the only consistency requirement on the structure of L is that $\{(y,x) \mid y \in S_x\}$ constitutes a partial order.

The second result states that every independence in a semi-graphoid can be inferred from at least one stratified protocol.

Theorem 4: If M is any semi-graphoid then the set of DAGs generated from all stratified protocols of M is a perfect map of M. (The criterion for separation relative to a set of DAGs is that d-separation must exist in at least one of the DAGs.)

Thus, even though every triplet in a stratified protocol asserts an independency relative to a singleton element; the sum total of such triplets is sufficient to encode all the set-to-set independencies embedded in the semi-graphoid.

Conclusions

This paper demonstrates that directed acyclic graphs (DAGs) possess powerful inferential properties. If an input set of dependencies is given in the form of a stratified protocol, then all implications of this input can be deduced efficiently, by graphical manipulations, instead of logical derivations.

No equivalent protocol of similar parsimony is known to work for undirected graphs, unless the generating model is a full graphoid, namely, unless logical, functional and definitional constraints are excluded from the model. Thus, DAGs appear to provide powerful inference tools for handling data dependencies of the type encountered in both probabilistic and logical reasoning. This feature helps explain the prevailing use of DAGs in causal models and semantic nets.

References

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

[Duda *et al.*, 1976] Richard.O. Duda, Peter.E. Hart and Nils.J. Nilsson. *Subjective Bayesian Methods for Rule-Based Inference Systems*, Proceedings, 1976 National Computer Conference (AFIPS Conference Proceedings), 45:1075-1082, 1976.

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

[Geiger, 1987] Daniel Geiger. *The Non-Axiomatizability of Dependencies in Directed Acyclic Graphs*, Technical Report R-83, Cognitive Systems Laboratory, UCLA, 1987.

[Kenny, 1979] David A. Kenny. *Correlation and Causality*. John Wiley and Sons, 1979.

[Lauritzen, 1982] S. L. Lauritzen. *Lectures on Contingency Tables*, University of Aalborg, Aalborg, Denmark, 1982.

[Montanari, 1974] Ugo Montanari. *Networks of Constraints*, Information Science, 7:95-132, 1974.

[Parker and Parsay, 1980] Stott Parker and Kamran Parsay. *Inferences Involving Embedded Multivalued Dependencies and Transitive Dependencies*, In Proceedings International Conference on Management of Data (ACM-SIGMOD), pages 52-57, 1980.

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

[Pearl and Paz, 1986] Judea Pearl and Azaria Paz. *GRAPHOIDS: A Graph-based Logic for Reasoning about Relevance Relations*, In Proceedings, ECAI-86, Brighton, United Kingdom, June 1986.

[Schank, 1972] Roger Schank. *Conceptual Dependency: A Theory of Natural Language Understanding*, Cognitive Psychology, 3(4), 1972.

[Shenoy and Shafer, 1986] Prakash P. Shenoy and Glen Shafer. *Propagating Belief Functions with Local Computations*" IEEE Expert, 1(3):43-52, 1986.

[Sowa, 1983] John F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, Reading, Massachusetts, 1983.

[Tarjan and Yannakakis, 1984] Robert. E. Tarjan and M. Yannakakis. *Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs and Selectively Reduce Acyclic Hypergraphs,*" SIAM J. Computing, 13:566-579, 1984.

[Verma, 1987] Thomas S. Verma. *Causal Networks: Semantics and Expressiveness*, Technical Report R-65, Cognitive Systems Laboratory, UCLA, 1987.

[Woods, 1975] William A. Woods. *What's in a Link? Foundations for Semantic Network*, Bobrow and Collins (Eds.), Representation and Understanding, Academic Press, 1975.