THE LOGIC OF REPRESENTING DEPENDENCIES BY DIRECTED GRAPHS

Judea Pearl
Thomas Verma

February 1987
CSD-870004

TECHNICAL REPORT
R-79
February, 1987

THE LOGIC OF REPRESENTING DEPENDENCIES BY DIRECTED GRAPHS

by

Judea Pearl & Thomas Verma

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

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 (PD), Embedded-Multi-Valued Dependencies (EMVD), Undirected Graphs (UG) 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 UGs, that DAGs completely represent the closure of their generating bases, and that they offer an effective computational device for testing consistency in a given set of dependencies as well as inferring new dependencies from that set. These properties might explain the prevailing use of DAGs in causal reasoning and semantic nets.

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 Grant #DCR 85-01234.

1. INTRODUCTION

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 relevant for assessing how long we are about to wait for the next bus. However, 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 allow assertions about dependency relationships to be expressed qualitatively, directly and explicitly. Moreover, the verification of dependencies should be accomplished swiftly 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. The former change will be called *monotonic* as it narrows the scope of propositions relevant to the target conclusion, and the latter will be called *nonmonotonic* as the scope of relevant propositions widens. For example, in trying to predict whether I am going to be late for a meeting, it is normally a good idea to ask somebody on the street for the time. However, once I establish the precise time by listening to the radio, asking people for the time becomes superfluous, and their responses would be irrelevant, thus demonstrating monotonic change of dependencies. For an example of a nonmonotonic relationship, consider the following: Normally, knowing the color of X's car tells me nothing about the color of Y's, but if X were to tell me that he almost mistook Y's car for his own, a new dependency is created between the two color variables -- whatever I learn about the color of X's car will have bearing on what I believe the color of Y's car to be. What logic would facilitate these two modes of reasoning?

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

$$P(x,y \mid z) = P(x \mid z)P(y \mid z) \tag{1}$$

Accordingly, if x and y are marginally dependent (i.e., dependent, when z is unknown) and become independent given z, a monotonic relationship holds. Conversely, if x and y are marginally independent and become dependent upon learning the value of z, a nonmonotonic relationship between x, y and z is captured. These relationships 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.

However, 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, not from manipulating numbers or tables. 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.

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 our earlier requirements of explicitness, saliency and stability, i.e., the links in the graph permit us to directly and categorically express 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 network.

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 is a perfect example of such a relational structure. 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. Thus, many different topologies might be used to display the set of dependencies embodied in P, each capturing a fragment of the set. We shall also 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.

Why Directed Graphs?

The main weakness of undirected graphs stems from their inability to represent nonmonotonic dependencies; two independent variables must be directly connected by an edge, merely because 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 $\rightarrow$ *bell* $\leftarrow$ *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 paper treats directed graphs as a language of expressing dependencies. Section 2 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 3 compares the expressive power of directed graphs to that of undirected graphs and shows the superiority of the former. Section 4, explores the power of directed graphs to cover data dependencies of the type produced by probabilistic or logical models. The main contribution 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 source of that power, and highlighting its limitations.

2. DEPENDENCY MODELS, MAPS ANd GRAPHOIDS

Definition: A *Dependency Model (DM)* 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

$$P(x \mid zy) = P(x \mid z) \tag{2}$$

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.

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 holds for every triplet (X, Z, Y). Thus, PD (and, similarly, PD$^-$, UGD, DAGD, SG, and G 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 and x_2 are any instantiations of the attributes in X, y_1 and y_2 are any instantiations of the attributes in Y and z is any instantiation of the attributes in Z, then by definition $I(X, Z, Y)_R$ iff

$$< x_1\, y_1\, z > \;\&\; < x_2\, y_2\, z > \;\Rightarrow\; < x_1\, y_2\, z > \tag{3}$$

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 the most general 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 (UG) G. If the function $A(n)$ denotes the set of nodes which are adjacent to n 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 cut-set 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 the function $C(n)$ denotes the set of nodes which are direct descendants of node n in G and the function $D(n)$ denotes the set of all n's descendants, then by definition $I(X, Z, Y)_G$ iff there is no sequence of nodes $\{\eta_i\}_1^n$ (with possible repetitions) for which all of the following hold:

$$\begin{aligned}
&1.\ \ \eta_1 \in X \\
&2.\ \ \eta_n \in Y \\
&3.\ \ \eta_i \in C(\eta_{i+1}) \ \textit{or}\ \ \eta_{i+1} \in C(\eta_i) \\
&4.\ \ \eta_i \in C(\eta_{i+1}) \cap C(\eta_{i-1}) \Rightarrow \eta_i \in Z \ \textit{or}\ \ D(\eta_i) \cap Z \neq \varnothing \\
&5.\ \ \eta_i \notin C(\eta_{i+1}) \cap C(\eta_{i-1}) \Rightarrow \eta_i \notin Z
\end{aligned} \tag{4}$$

In other words, parts 1-5 forbid the existence of a bi-directed path starting from a node in X and ending at a node in Y (part 1-3) along which every node with converging arrows either is or has a descendent in Z (part 4) and every other node is outside Z (part 5).

This criterion was called *d-separation* in [Pearl, 1986]; part 5 corresponds to ordinary vertex separation in undirected graphs while part 4 conveys the idea that the inputs of any causal mechanism become dependent once the output is known. 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 both paths between 2 and 3 are "blocked". However X and Y are not separated by $Z' = \{1, 5\}$ because node 5, as a descendent of 4, "unblocks" the connection between the converging arrows at 4, thus opening a pathway between 2 and 3.

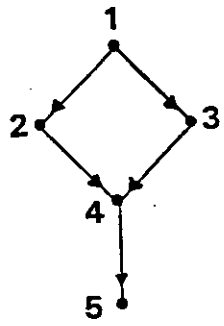

Figure 1

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

It is straight forward to show that all of the specialized 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 dependencies 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, 1981], and a similar result has recently been reported for DAGD [Geiger, 1987]. PD is conjectured to be equivalent to SD (i.e., $M \in PD <=> M \in SG$) but no proof (nor disproof) is in sight.

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

Intersection: $(X, ZY, W)\ \&\ (X, ZW, Y) \in M \Rightarrow (X, Z, YW) \in M$ (6)

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

The most important properties of graphoids [Pearl and Paz, 1986] is that they posses unique 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 3, we will show that the use of DAG's provides a local construction of I-maps for every semi-graphoid.

Definition: Let M be a dependency model from some class $\boldsymbol{M}$ of dependency models. A subset $B \subseteq M$ of triplets is a *$\boldsymbol{M}$–basis* of M iff every model $M' \in \boldsymbol{M}$ which contains B also contains M. Thus, a basis provides a complete encoding of the information contained in M; knowing B and $\boldsymbol{M}$ 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:

$$B_N = \{(x, N(x), U - x - N(x)), \forall\, x \in V\} \tag{7}$$

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 set:

$$B = \{(x_i, PA(x_i), \{x_1, \ldots, x_{i-1}\} - PA(x_i)),\ \ i = 1, \ldots, n\} \tag{8}$$

B is a DAG-basis of G since the closure of B coincides with the independencies displayed by G.

3. 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-monotonic and 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 answer to the first question is, clearly, negative. For instance, the dependency structure of the diamond-shaped graph of Fig.2(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 nonchordal graphs (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 UGs 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 2(a). Introducing an auxiliary variable E in the manner shown in Figure 1(b) creates a DAG model on five variables which also as-

serts $I(C, B, D)$.

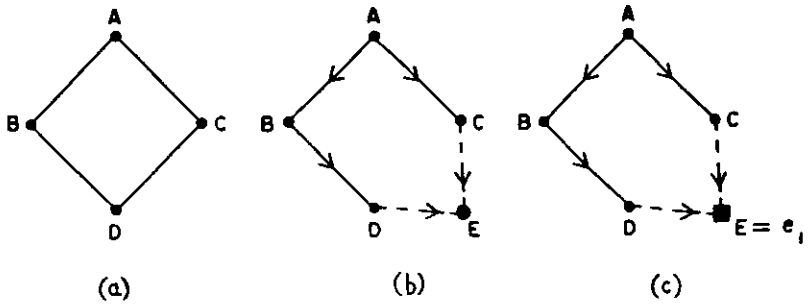

Figure 2

If we "clamp" the auxiliary variable E at some fixed value $E = e_1$, as in Figure 2(c), the dependency structure projected on A, B, C, D is identical to the original structure of Figure 2(a), i.e., $I(A, BC, D)$ and $I(B, AD, C)$.

In general, one can show that the introduction of auxiliary variables renders the DAG representation more powerful than undirected graphs:

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

Another method of improving the expressive power of DAGD (without introducing auxiliary nodes) is to permit both directed and undirected links (but no directed cycles) and use a separation criterion similar to that of DAGs. One can show, indeed, that the expressive power of hybrid graphs is greater than UGD and DAGD combined (Verma, 1987).

4. HOW FAITHFUL ARE DAG'S?

Suppose someone (e.g., an expert) provides us with a list L of triplets representing a set of independencies in some (undisclosed) dependency model M, of a known type. 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 contains 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 recursive protocol L_θ of a dependency model M is any set of pairs

$$L_\theta = \{(x, S_x) \mid \forall\ x \in U\} \tag{9}$$

such that

$$(x, S_x) \in L_\theta \Rightarrow (x, S_x, U_{\theta(x)} - S_x) \in M \tag{10}$$

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.

Recursive 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 results 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 recursive protocol L_θ of M is an I-map of M.

Corrollary: If L_θ is any recursive protocol of some dependency model M, the DAG generated from L_θ is a perfect map of the semi-graphoid closure of L_θ.

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 4 holds for any generating order. Note also that the internal consistency of any recursive protocol can be verified by simply testing whether $\{(y, x) | y \in S_x\}$ constitutes a partial order.

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

Theorem 4: If M is any semi-graphoid then the set of DAGs generated from all recursive protocols of B is a perfect map of M if the criterion for separation is that d-separation must exist in one of the DAGs.

CONCLUSIONS

This paper demonstrates that directed acyclic graphs (DAGs) possess powerful inferential properties. If the input set of dependencies is given in the form of a recursive 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 logical reasoning and truth maintenance systems. This feature help explain the prevailing use of DAGs in causal reasoning and semantic nets.

REFERENCES

R. O. Duda, P. E. Hart & N. J. Nilsson, "Subjective Bayesian Methods for Rule-Based Inference Systems," *Proceedings, 1976 National Computer Conference (AFIPS Conference Proceedings)*, 45, 1075-1082, 1976.

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

U. Montanari, "Networks of Constraints," *Information Science*, Vol. 7, 1974, pp.95-132.

S. Parker and K. Parsay, "Inferences Involving Embedded Multivalued Dependencies and Transitive Dependencies," *Proc. International Conf. on Management of Data* (ACM-SIGMOD), pp.52-57, 1980.

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

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)*.

R. Schank, "Conceptual Dependency: a Theory of Natural Language Understanding," *Cognitive Psychology*, Vol. 3, No. (4), 1972.

P. Shenoy and G. Shafer, "Propagating Belief Functions with Local Computations" *IEEE Expert* 1, (3), pp. 43-52, (1986).

J. F. Sowa, "Conceptual Structures: Information Processing in Mind and Machine," Addison-Wesley, Reading, Mass. (1983).

Tarjan & M. Yannakakis, "Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs and Selectively Reduce Acyclic Hypergraphs," *SIAM J. Computing*, Vol. 13, 1984, pp.566-579.

T. S. Verma, "Causal Networks: Semantics and Expressiveness," UCLA Cognitive Systems Laboratory *Technical Report (R-65)*, Los Angeles, California, 1987.

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