Computer Science Department Technical Report
Cognitive Systems Laboratory
University of California
Los Angeles, CA 90024-1596

IDENTIFYING INDEPENDENCE IN BAYESIAN NETWORKS

Dan Geiger
Thomas Verma
Judea Pearl

June 1989
CSD-890028

TECHNICAL REPORT
R-116
October 1988

IDENTIFYING INDEPENDENCE IN BAYESIAN NETWORKS *

Dan Geiger, Thomas Verma & Judea Pearl
Cognitive Systems Laboratory, Computer Science Department
University of California Los Angeles, CA 90024
Net address: geiger@cs.ucla.edu
Net address: verma@cs.ucla.edu
Net address: judea@cs.ucla.edu

ABSTRACT

In this paper we explore the role of conditional independence in the theory of Bayesian Networks. We show that conditional independence is the key for identifying what information is irrelevant and which parameters are immaterial for performing a given computation. We propose an algorithm based on a graphical criterion, d-separation, that satisfies all independencies encoded in a Bayesian Network. The correctness and optimality of our algorithm is implied by the *soundness* and *completeness* of d-separation with respect to probability theory. Finally, we define an enhanced version of d-separation, called D-separation, which extends our algorithms to networks that encode functional dependencies.

Key words: Bayesian Networks, Conditional independence, Influence Diagrams, Graphoids, Perfect maps, Probabilistic reasoning.

*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)".

1. Introduction

Networks employing Directed Acyclic Graphs (DAGs) are common representation schemes for probabilistic knowledge. Their usage is spread among various disciples such as: Artificial intelligence, decision analysis, economics and statistics, each of which adopt their own name bayesian belief networks, causal networks, recursive models, probabilistic influence diagrams (PID) and more [, , , , , , ,]. In this paper we adopt the name Bayesian Network and DAGs interchangingly. A Bayesian Network is a graphical encoding of a distribution via a DAG. Each node i in a Bayesian Network corresponds to a variable x_i and each node is regarded as a storage cell for the distribution $P(x_i \mid X_{P(i)})$ where $X_{P(i)}$ is a set of variables that correspond to the parent nodes of i, denoted $P(i)$. The distribution represented by a Bayesian Network is composed via

$$P(x_1, \cdots, x_n) = \prod_{i=1}^{n} P(x_i \mid X_{P(i)})$$

(when i has no parents $X_{P(i)} = 0$). The role of a Bayesian Network is to answer queries such as the believe in a proposition $x_i = true$ given the values of an observed set of variables X_J, or, such as the most probable explanation for an observation. In this paper we concentrate on qualitative non-numeric properties underlying query-answering mechanisms in Bayesian Networks. We answer the following two questions:

1. Given a task to compute $P(x_i \mid X_J)$ and a variable x_k, and without resorting to numeric parameters, is the answer to the query sensitive to the value of x_k?

2. Is the answer sensitive to the *parameters* $P(x_k \mid X_{P(n)})$ stored in node k?

The answer to Question 1 is obvious; the value of x_k would not affect the query $P(x_i \mid X_J)$ if $P(x_i \mid X_J) = P(x_i \mid X_J, x_k)$ for all values of x_i, x_k and X_J. This equality denoted by $I(x_i, X_J, x_k)$ is a *(conditional independence) statement* stating that x_i is independent of x_k, given X_J. Our paper emphasizes the importance of these statements in formalizing and manipulating Bayesian Networks. We provide two graphical criterions d-separation and D-separation that identify in linear time each and every independency statement that is implied by the topology of the network and reveals which portions are relevant to the task at hand. The D-separation criteria is an enhancement of d-separation for the case where deterministic variables are present, namely, when the value of a variable x_i is a deterministic function of $X_{P(i)}$. Such a node is depicted in a DAG by a double circle node and its presense imposes additional independence assumptions for the underlying distribution.

The answer to the second question is also given (in Section 3) in terms of separation. Its relation to the first problem is discussed. Finally, we compare our solution to this problem to the one suggested by Shachter [1988].

The rest of the paper is organized as follows: Section 2 provides the definition of d-separation, its soundness and completeness. Section 3 provides a linear algorithm to identify independencies in Bayesian Networks. Its correctness and optimality are proven. Section 4 extends the results of previous sections to Bayesian Networks that contain deterministic nodes. Section 5 contains the soundness and completeness proofs.

2. Soundness and Completeness of d-separation

The definition of d-separation is best motivated by regarding DAGs as a representation of causal relationships. Designating a node for every variable and assigning a link between every cause to

each of its direct consequences defines a graphical representation of a causal hierarchy. For example, the propositions "It is raining" (α), "the pavement is wet" (β) and "John slipped on the pavement" (γ) are well represented by a three node chain, from α through β to γ; it indicates that either rain or wet pavement could cause slipping, yet wet pavement is designated as the *direct cause;* rain could cause someone to slip if it wets the pavement, but not if the pavement is covered. Moreover, knowing the condition of the pavement renders "slipping" and "raining" independent, and this is represented graphically by a d-separation condition, $I(\alpha, \gamma, \beta)_D$, showing node α and β separated from each other by node γ. Assume that "broken pipe" (δ) is considered another direct cause for wet pavement, as in figure 1. An induced dependency exists between the two events that may cause the pavement to get wet: "rain" and "broken pipe". Although they appear connected in Figure 1, these propositions are marginally independent and become dependent once we learn that the pavement is wet or that someone broke his leg. An increase in our belief in either cause would decrease our belief in the other as it would "explain away" the observation. The following definition of d-separation permits us to graphically identify such induced dependencies from the DAG (d connoted "directional").

Definition: If X, Y, and Z are three disjoint subsets of nodes in a DAG D, then Z is said to d-separate X from Y, denoted $I(X, Z, Y)_D$, iff there is no path* from a node in X to a node in Y along which every node that delivers an arrow is outside Z and every node with converging arrows either is or has a descendant in Z. A path satisfying the conditions above is said to be *active,* otherwise it is said to be *blocked* (by Z). Whenever a *statement* $I(X, Z, Y)_D$ holds in a DAG D, the predicate $I(X, Z, Y)$ is said to be *graphically-verified* (or an *independency),* otherwise it is *graphically-unverified* by D (or a *dependency).*

* By *path* we mean a sequence of edges in the underlying undirected graph, i.e ignoring the directionality of the links.

In figure 2, for example, $X = \{2\}$ and $Y = \{3\}$ are d-separated by $Z = \{1\}$; the path $2 \leftarrow 1 \rightarrow 3$ is blocked by $1 \in Z$ while the path $2 \rightarrow 4 \leftarrow 3$ is blocked because 4 and all its descendents are outside Z. Thus $I(2, 1, 3)$ is graphically-verified by D. However, X and Y are not d- separated by $Z' = \{1, 5\}$ because the path $2 \rightarrow 4 \leftarrow 3$ is rendered active: learning the value of the consequence 5, renders its causes 2 and 3 dependent, like opening a pathway along the converging arrows at 4. Consequently, $I(2, \{1,5\},3)$ does not hold and is therefore graphically-unverified by D.

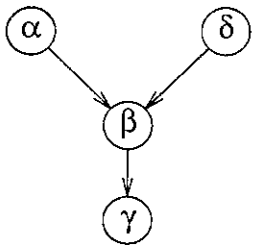

Figure 1

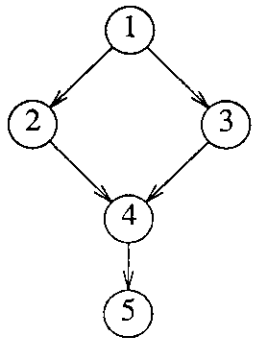

Figure 2

Definition: If X, Y, and Z are three disjoint subsets of variables of a distribution P, then X and Y are said to be conditionally independent given Z, denoted $I(X, Z, Y)_P$ iff $P(X \mid Z, Y) = P(X \mid Z)$ for all possible values of X, Y and Z. $I(X, Z, Y)_P$ is called a *(conditional independence) statement.* A conditional independence statement σ *logically follows* from a set Σ of such statements if σ holds in every distribution that obeys Σ, in which case we also say that σ is a *valid consequence* of Σ.

Ideally, to employ a DAG D as a graphical representation for dependencies of some distribution P we would like to require that for every three disjoint sets of variables in P (and nodes in D) the following equivalence would hold

$$I(X, Z, Y)_D \quad \textit{iff} \quad I(X, Z, Y)_P \tag{2}$$

This would provide a clear graphical representation of all variables that are conditionally independent. When equation (2) holds, D is said to be a *perfect map* of P. Unfortunately, this requirement is often to strong because there are many distributions that have no perfect map in DAGs. The spectrum of probabilistic dependencies is in fact so rich that it cannot be cast into any representation scheme that uses polynomial amount of storage ([Verma, 1987]). Geiger [1987] provides a graphical representation based on a collection of graphs (Multi-DAGs) that is powerful enough to perfectly represent an arbitrary distribution, however, as shown by Verma, it requires, on the average, an exponential number of DAGs. Being unable to provide perfect maps at a reasonable cost, we compromise the requirement that the graphs represent each and every dependency of P, and allow some independencies to escape representation.

Definition: A DAG D is said to be an *I-map* of P if for every three disjoint subsets X, Y and Z of variables the following holds:

$$I(X, Z, Y)_D \quad \Rightarrow \quad I(X, Z, Y)_P$$

The natural requirement for these I-maps is that the number of undisplayed independencies be minimized.

The task of finding a DAG which is a minimal I-map of a given distribution P was solved in [Pearl & Verma, 1987]. Their algorithm consists of the following steps: assign a total ordering d to the variables of P. For each variable x_i of P, identify a minimal set of predecessors X_{S_i} that renders x_i independent of all its other predecessors (in the ordering of the first step). Assign a direct link from every node in S_i to i. The resulting DAG is an I-map of P, and is minimal in the sense that no edge can be deleted without destroying its I-mapness. The input list

L for this construction consists of n conditional independence statements, one for each variable, all of the form $I(x_i, X_{S_i}, X_{U_{(i)}}-X_{S_i})$ where $X_{U_{(i)}}$ is the set of predecessors of x_i and X_{S_i} is a subset of $X_{U_{(i)}}$ that renders x_i conditionally independent of all its other predecessors. This set of conditional independence statements is called a *causal* input list and is said to *define* the DAG D. The term "causal" input list is derived from the following analogy: Suppose we order the variables chronologically, such that a cause always precedes its effect. Then, from all potential causes of an effect i, a causal input list selects a minimal subset that is sufficient to explain i, thus rendering all other preceding events superfluous. This selected subset of variables are considered the *direct causes* of i and therefore each is connected to it by a direct link.

Clearly, the constructed DAG represents more independencies than those listed in the input, namely, all those that are graphically verified by the d-separation criterion. [Pearl & Verma, 1987] analysis guarantees that all graphically-verified statements are indeed valid in P i.e., the DAG is an I-map of P. However, this paper shows that the constructed DAG has an additional property; it graphically-verifies **every** conditional independence statement that logically follows from L (i.e. holds in every distribution that obeys L). Hence, we cannot hope to improve the d-separation criterion to display more independencies, because all valid consequences of L (which defines D) are already captured by d-separation.

The three theorems below formalize the above discussion.

Theorem 1 (soundness) [Pearl & Verma, 1987]: Let D be a DAG defined by a causal input list L. Then, every graphically-verified statement is a valid consequence of L.

Theorem 2 (closure) [Pearl & Verma, 1987]: Let D be a DAG defined by a causal input list L. Then, the set of graphically-verified statements is exactly the closure of L under axioms (1.a) through (1.d).

Theorem 3 (completeness) [Geiger & Pearl, 1988]: Let D be a DAG defined by a causal input list L. Then, every valid consequence of L is graphically-verified by D (equivalently, every graphically-unverified statement in D is not a valid consequence of L).

Theorem 1 guarantees that a DAG displays only valid statements. Theorem 2 guarantees that a DAG displays all statements that are derivable from L via axioms (1). The third theorem assures a DAG displays all statements that logically follow from L i.e., the axioms in (1) are complete, capable of deriving all valid consequences of a causal input list. Moreover, since a statement in a DAG can be verified in linear time, theorem 3 provides a complete polynomial inference mechanism for deriving all independency statements that are implied by a causal input set. A generalized version of these theorems is proven in Section 5.

The first two theorems are more general than the third in the sense that they hold for every dependence relationship that obeys axioms (1.a) through (1.d), not necessarily those based on probabilistic conditional independence (see proof in Section 5). Among these dependence relationships are partial correlations ([Pearl & Paz, 1986]) and qualitative dependencies ([Fagin, 1982], [Shafer at al, 1987]) which can readily be shown to obey axioms (1). Thus, for example, the transformation of arc-reversal and node removal ([Howard & Matheson, 1981; Olmsted, 1983]) can be shown valid by purely graphical consideration, simply showing that every statement verified in the transformed graph is also graphically-verified in the original graph.

We conclude this section by showing how these theorems can be employed as an inference mechanism. Assume an expert has identified the following conditional independencies between variables denoted 1 through 5:

$$L = \{ I(2, 1, \varnothing),\ I(3, 1, 2),\ I(4, 23, 1),\ I(5, 4, 123) \}$$

(the first statement in L is trivial). We raise two questions. First, what is the set of all valid

consequences of L ? Second, in particular, is $I(3, 124, 5)$ a valid consequence of L ? For general input lists the answer for such questions may be undecidable but, since L is a causal list, it defines a DAG that graphically verifies each and every valid consequences of L. The DAG D is the one shown in figure 2. Therefore, the DAG constitutes a dense representation of all valid consequences of L. To answer the second question, we simply observe that $I(3, 124, 5)$ is graphically-verified in D. A graph-based algorithm for another subclass of statements, called *fixed context* statements, is given in [Geiger & Pearl, 1988]. In that paper, results analogous to theorem 1 through 3 are proven for Markov-fields; a representation scheme based on undirected graphs ([Isham, 1981], [Lauritzen, 1982]).

3. A Linear Algorithm for Identifying Independencies.

The algorithm below finds the set of all nodes Y that are not d-separated from X given Z. Accordingly, a statement $I(X, Z, Y')_D$ holds in D iff $Y \cap Y' = \varnothing$, therefore the algorithm can be used to evaluate an arbitrary d-separation statement, in linear time. The algorithm is a variant of depth first search; it employs a stack to keep the current node and it marks visited links. The following notations are used: E for the set of edges of D, $u-v$ for an edge between u and v (regardless the direction) and Descendent_in_Z[] for a boolean array indicating for each node if it has a descendent in Z.

For all $x \in X$ *do* $E := E \cup \{x \rightarrow x_0\}$

For all $z \in Z$ *do* *Descendent_in_Z[z]:=true*

$Y := \varnothing$

push $(STACK, x_0)$

While $STACK \neq \varnothing$

v := *top* $(STACK)$

u := *one_below_top* $(STACK)$

If v *has an unmarked link (outgoing links selected first) to* w *such that*

either [v *is a head-to-head node on* $u-v-w$ *and Descendent_in_Z(v)]*

or [v *is not a head-to-head node on* $u-v-w$ *and* $v \notin Z$ *]*

then

mark$(v-w)$:= *true*

push$(STACK, w)$

$Y := Y \cup \{ w \}$

else

pop(v)

If Descendent_in_Z $[v]$ & $(u \rightarrow v) \in E$ *then* *Descendent_in_Z* $[u]$:= *true*

end

Proposition 1: Let $p = (s_0 = \alpha, s_1, \cdots, s_n = \beta)$ be a path between α and β that is active by a set of nodes J. Then p remains active under $J \cup J'$ where J' is any set of nodes satisfying $J' \cap \{s_1, \cdots, s_n\} = \varnothing$.

Proof: 1. Any node on p with converging arrows is or has a descendent in J and therefore also in $J \cup J'$. 2. Any other node s_i on p is not in $J \cup J'$ because $J' \cap \{s_1, \cdots, s_n\} = \varnothing$ and because p is active by J.

Definition: Let J_v be $\{ j \mid j \in J$ and j *is placed on the stack before* $v \}$ (if v is not placed on the stack, J_v contains all $j \in J$ that are placed on the stack during the entire execution).

Lemma: For every node w not in $\{j_0 \cup J\}$, w is placed on the stack iff there is an active path by J_w between v_0 and w.

Only if: Proof by induction on the size of the stack when w is first added to it. For $|STACK| = 1$, the stack contains only v_0 which is a member of $\{j_0 \cup J\}$, thus the base case trivially holds. let v be the top element on the stack when w is first placed on the stack. By the induction hypothesis, there is a path, $(s_0 = j_0, \cdots, s_{k-1} = u, s_k = v)$ between j_0 and v that is active by J_v.

If $v \in J$ then w could have been added to the stack only if v has converging arrows on the path $p_w = (s_0, \cdots, u, v, w)$. This path is active by $J_v \cup \{ v \}$ because 1. every h-h node on p_w between v_0 and v is or has a descendent in $J_v \subseteq J_w$ and v itself is a member of J and placed on the stack before w, hence, by definition, $v \in J_w$. 2. Every other node on p_w is not in $J_v \subseteq J_w$.

By proposition 1, the path p_w is also active by J_w because $J_w - J_v$ does not intersect p_w.

4. Networks with Deterministic Nodes.

The completeness of d-separation (theorem 3) assumes that the input list L is causal, containing only statements of the form $I(i, S_i, U_{(i)}-S_i)$. Occasionly, however, we are in possession of stronger forms of independence relationships, in which case additional statements should be read of the DAG. A common example is the case of a variable that is functionally dependent on its corresponding parents in the DAG (*deterministic variable,* [Shachter, 1988]). The existence of each such variable i could be encoded in L by a statement of *global* independence $I(i, S_i, U-S_i-i)$ asserting that conditioned on S_i, i is independent of all other variables, not merely of its predecessors. The independencies that are implied by the modified input list can be read from the DAG using an enhanced version of d-separation, named, D-separation.

Definition: If X, Y, and Z are three disjoint subsets of nodes in a DAG D, then Z is said to *D-separate* X from Y, iff there is no path from a node in X to a node in Y along which 1. every node which delivers an arrow is outside Z, 2. every node with converging arrows either is or has a descendant in Z and 3. no node is functionally determined by Z.

Definition: A node α is *functionally dependent* on Z iff it is a deterministic node and all its parents are functionally dependent on Z. If α is a deterministic node with no parents then it is functionally dependent on z.

The new criterion certifies all independencies that are revealed by d-separation plus additional ones due to the enhancement of the input list. Parallel to the discussion of Section 2, the following soundness and completeness results hold. Proofs are postponed to Section 5.

Theorem 1 (soundness): Let D be a DAG defined by an enhanced causal list L. Then, every graphically-verified statement is a valid consequence of L.

Theorem 2 (closure): Let D be a DAG defined by an enhanced causal list L. Then, the set of graphically-verified statements is exactly the closure of L under axioms (1.a) through (1.d).

Theorem 3 (completeness): Let D be a DAG defined by an enhanced causal list L. Then, every valid consequence of L is graphically-verified by D (equivalently, every graphically-unverified statement in D is not a valid consequence of L).

These graphical criteria provide easy means of recognizing conditional independence in influence diagrams as well as identifying the set of parameters needed for any given computation. Shachter [1988] has devised an algorithm for finding a set of nodes M guaranteed to contain sufficient information for computing $P(x \mid y)$, for two arbitrary sets of variables x and y. The outcome of Shachter's algorithm can now be stated declaratively; M contains every ancestor of $x \cup y$ that is not D-separated from x given y and none other. The completeness of D-separation implies that M is minimal; no node in M can be excluded on purely topological grounds (i.e., without considering the numerical values of the probabilities involved).

5. Proofs.