DECIDING CONSISTENCY IN INHERITANCE NETWORKS

Judea Pearl

September 1987
CSD-870053

Technical Report
R-96
August 1987

DECIDING CONSISTENCY IN INHERITANCE NETWORKS*

Judea Pearl
UCLA Computer Science Department
Cognitive Systems Laboratory
Los Angeles, CA. 90024-1596

ABSTRACT

An *Inheritance Network* is a labeled directed graph, $\Gamma = (V, E^+ \cup E^-)$, where the vertices V correspond to a set of atomic properties and E^+ and E^- are sets of positive and negative arcs, respectively. Positive arcs correspond to default rules of the type "p is a q" (e.g., "penguins are birds," "birds fly") and negative arcs correspond to default rules of the type "p is not a q" (e.g., "penguins do not fly").

An inheritance network is said to be *consistent* if there exists a probability model in which all the defaults are highly probable. This report establishes the following criterion for deciding consistency: Γ is consistent iff for every pair of conflicting arcs $(p_1, q) \in E^-$ and $(p_2, q) \in E^+$, p_1 and p_2 must be distinct and there is no cycle of positive arcs that embraces both p_1 and p_2.

* This work was supported in part by the National Science Foundation Grant, IRI-8610155

Technical Report

R-96

August 1987

DECIDING CONSISTENCY IN INHERITANCE NETWORKS

Judea Pearl

1. INTRODUCTION

An *Inheritance Network* is a labeled directed graph, $\Gamma = (V, E^+ \cup E^-)$, where the vertices V correspond to a set of atomic properties and E^+ and E^- are sets of positive and negative arcs, respectively. Positive arcs correspond to default rules of the type "p is a q" (e.g., "penguins are birds," "birds fly") and negative arcs correspond to default rules of the type "p is not a q" (e.g., "penguins do not fly").

An inheritance network is said to be *consistent* if there exists a probability model in which all the defaults are highly probable (Adams 1966, Pearl 1987). Formally, Γ is *consistent* if, for any $\varepsilon > 0$, there exists a probability function P such that:

$$P(q \mid p) \geq 1 - \varepsilon \quad \textit{whenever} \quad (p, q) \in E^+$$

$$P(q \mid p) \leq \varepsilon \quad \textit{whenever} \quad (p, q) \in E^-$$

Thus, consistent networks represent sets of default rules where conflicts can be explained away by appealing to rare exceptions, while inconsistent networks represent inexcusable contradic-

tions, usually due to sloppy design or thoughtless inputs.

This report establishes the following criterion for deciding consistency: Γ is consistent iff for every pair of conflicting arcs $(p_1, q) \in E^-$ and $(p_2, q) \in E^+$, p_1 and p_2 must be distinct and there is no cycle of positive arcs that embraces both p_1 and p_2. The proof of this criterion is based on a theorem by Adams (1966) which provides a decision procedure for probabilistic consistency of general sets of formula.

Def - An assignment t of $\{0, 1\}$ values to the vertices of Γ is said to *falsify* a positive arc $(x, y) \in E^+$ if $x=1$ and $y=0$ under t, and to *verify* $(x, y) \in E^+$ if $x=1, y=1$ under t. A negative arc $(x, y) \in E^-$ is falsified and verified by the respective assignments (1, 1) and (1, 0).

Def - A set of arcs $E' \subseteq E^+ \cup E^-$ is said to be *confirmed* by a particular 0–1 assignment t, if no arc in E' is falsified and at least one arc in E' is verified under t.

Theorem (Adams, 1966) - Γ is *consistent* if every non-empty subset of $E = E^+ \cup E^-$ is confirmed by some assignment t.

Given a network Γ, our task is to devise a graph-theoretic criterion for deciding if Γ is consistent. Theorem 1 establishes such a criterion for the case where the positive arcs of Γ do not form a cycle while Theorem 2 extends the criterion to general networks.

2. CONSISTENCY IN ACYCLIC NETWORKS

Theorem 1: If $\Gamma^+ = (V, E^+)$ is acyclic, then Γ is consistent iff it is locally consistent, i.e., it does not contain conflicting pairs, (e.g., $p \rightarrow q \;\; \& \;\; p \rightarrow \neg q$), nor self loops (e.g., $p \rightarrow \neg p$).

Proof: The proof is by induction along any total order consistent with the directions of the positive arcs E^+. We will assume that the theorem holds for any Γ' and show that it is always possible to add a new vertex y to Γ' without perturbing consistency, as long as y does not sprout positive arcs back to Γ'.

We need to prove that for any subset of arcs S which contains y, there exists a truth-value assignment t which confirms S (if S does not contain y, it is trivially confirmed, by the induction hypothesis). Let $\Gamma_S = (V_S, E_S^+ \cup E_S^-)$ be the subgraph induced by the arcs of S and Γ_{S-y} the corresponding graph with y and all arcs incident to y removed. Let $P_y = P_y^+ \cup P_y^-$ and $C_y = C_y^+ \cup C_y^-$ stand, respectively, for the sets of parents and children of y in Γ_S, with the superscript indicating the type of arcs involved. Acyclicity dictates $C_y^+ = \varnothing$ while the requirement of no conflicting pairs means $x \in P_y^+ \Longrightarrow x \notin P_y^-$.

The cases where either $P_y^+ = \varnothing$ or $C_y^- \neq \varnothing$ can be proven immediately. If $P_y^+ = \varnothing$ then the assignment $t' = t \cup \{y = 0\}$ obviously confirms S, where t is any assignment that confirms $S - y$ by the induction hypothesis. If $C_y^- \neq \varnothing$ we can use the trivial assignment $t = \{y = 1, x = 0, \forall x \in V_S - y\}$, which confirms S because the negative arcs emanating from y are verified and no other arc can be falsified.

It remains to consider only conditions where $P_y^+ \neq \varnothing$ and $C_y = \varnothing$ and these require the separate examination of several cases.

Case - 1: $P_y^- = \varnothing$, i.e., S contains no negative arcs pointing to y.

Define the following two sets of nodes in V_{S-y}:

A_1: nodes connected by negative arcs to at least one of their positive descendants in Γ^+_{S-y}.

A_2: nodes connected by negative arcs toward some non-descendant in Γ^+_{S-y}.

Definition: For any set Q of nodes, we shall say that a node x is a *youngest Q-node* if $x \in V^+_{S-y} \cap Q$ and no other (positive) descendant of x is in $V^+_{S-y} \cap Q$.

Let x be the youngest $P_y^+ \cup A_1 \cup A_2 - node$

Case - 1.1: x is in A_1, i.e., x has a descendant z in Γ^+_{S-y} and $(x, z) \in E_S^-$, as in Figure 1.

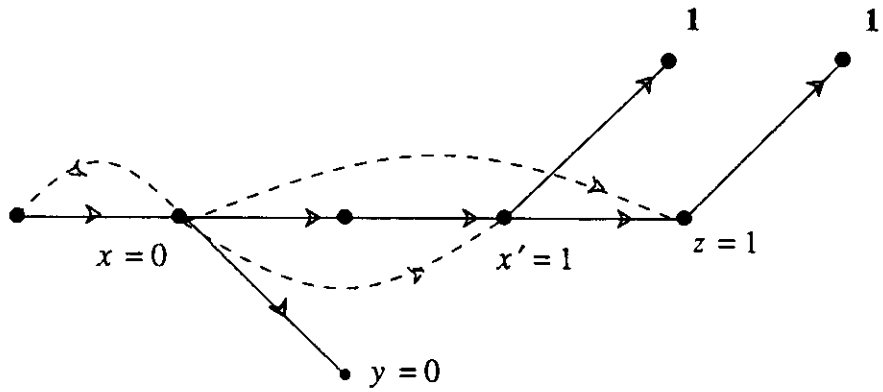

Figure 1

z cannot be a direct positive child of x because conflicting pairs are prohibited. Thus, there exists another node x' on a positive path between x and z such that $(x', z) \in E_S^+$. Let t assign 1 to x' and all its positive descendants in Γ^+_{S-y} and 0 to any other node in Γ. t confirms S because all the positive arcs at the subnetwork rooted at x' are verified while no arc is falsified -- the assumption that x is the youngest $P_y^+ \cup A_1 \cup A_2 - node$, precludes any 1-assigned node from

sprouting negative arcs.

Case - 1.2: x is not in A_1, and, since $P_y^+ \neq \varnothing$, x must be in $P_y^+ \cup A_2$ (see Figure 2). Thus, either

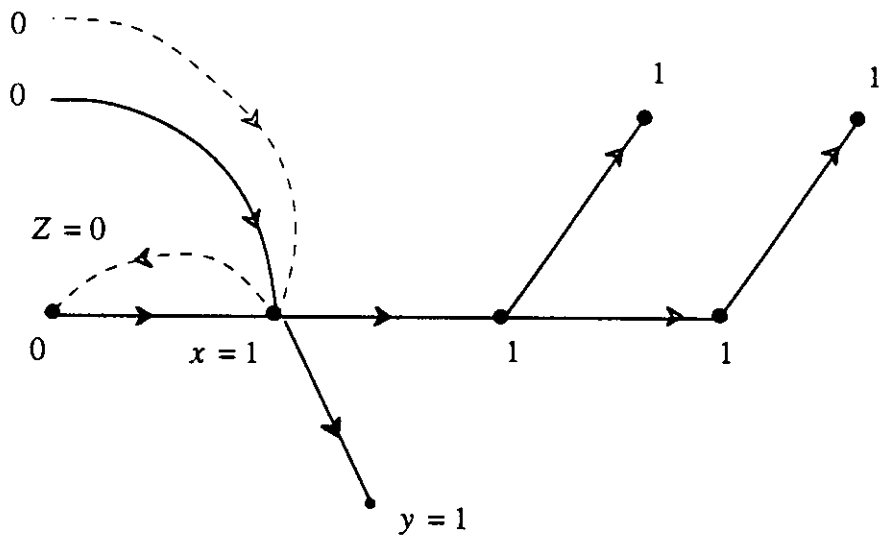

Figure 2

y receives a positive arc from x or there exists a non-descendant Z receiving a negative arc from x. Let t assign 1 to x, y and all positive descendants of x and 0 to all other nodes in Γ_{S-y}. t confirms S because there must be at least one verified arc sprouting from x while no arc is falsified -- no positive descendant of x sprouts any negative arc and those sprouting from x are verified, ending up at nodes assigned 0's.

Case - 2: $P_y^- \neq \varnothing$, i.e., S contains negative arcs pointing to y.

Again, let x be the youngest $P_y^+ \cup A_1 \cup A_2$ node.

Case - 2.1: $x \in A_1$

The assignment of case 1.1 still confirms S, because $y = 0$ prevents any negative arc pointing to y from being falsified.

Case 2.2: $x \notin A_1, x \in P_y^+ \cup A_2$.

The assignment of case 1.2 would still confirm S unless any of the 1-assigned descendants of x, say x', is also in P_y^- (see Figure 3). If this happens,

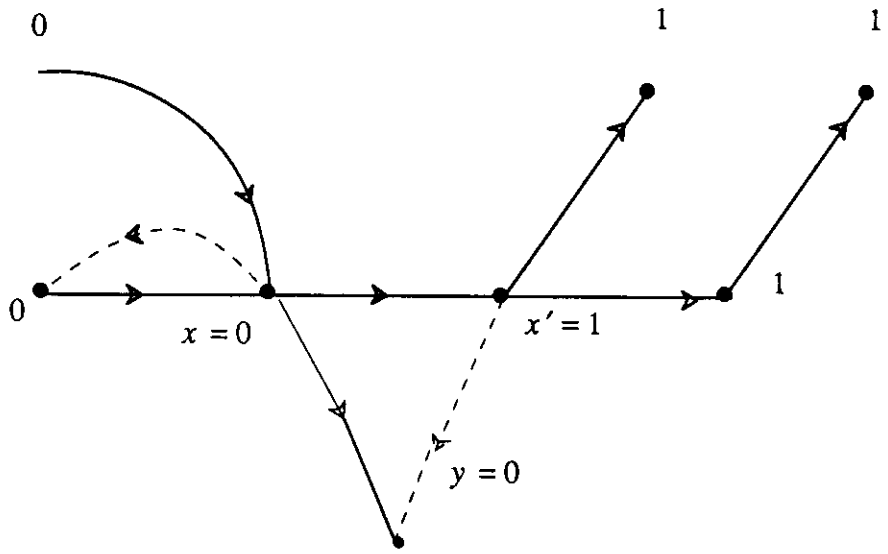

Figure 3

assign 1 only to x' and its positive descendants, assign 0 to all other nodes including x and y.

Case 2.3: $P_y^+ \cup A_1 \cup A_2 = \varnothing$

Let x' be the youngest P_y^- node. Assign 1 to x' and all its positive descendants, assign 0 to all other nodes, including y. Q.E.D.

3. CONSISTENCY IN GENERAL NETWORKS

Definition: A *positive cycle* of Γ is a set of nodes such that there exists a sequence of positive arcs connecting them in a cycle. A *cycle-cluster* is a maximal set of positive cycles such that any two cycles share at least one node. For example, the cycle-clusters in the network of Figure 4 are $\{a, b, c\}$ and $\{d, e, f, h, g\}$; the two cycles $\{d, e, f\}$ and $\{f, h, g\}$ are clustered together because they share node f.

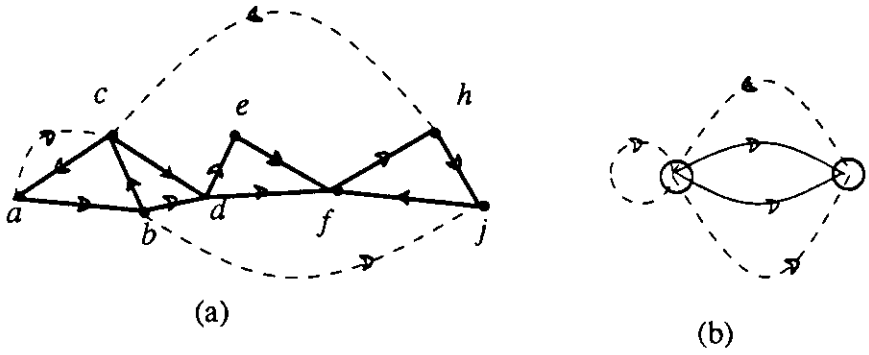

Figure 4

Definition: The cycle kernel of Γ is a subgraph $\Gamma_K = \{V_K, E_K^+ \cup E_K^-\}$ having the cycle clusters of Γ as vertices and a subset of E as arcs. An arc (X, Y) is in E_K^+ if Γ contains a positive arc between some node in cluster X and a node in cluster Y, and $X \neq Y$. An arc (X, Y) is in E_K^- if there is a negative arc between some node in X and a node in Y, including self loops (in case $X = Y$). In other words, Γ_K retains the original arcs of Γ with the exception of the positive arcs that make up the cycle clusters.

In the sequel we will use the terms clusters and kernel to denote cycle cluster and cycle kernel, respectively. Figure 4(b) illustrates the kernel of the network in Figure 4(a). Clearly, if

the positive subgraph of Γ is acyclic, then the kernel Γ_K is Γ itself. Also, the subgraph induced by the positive arcs of any kernel is acyclic. The reason is that any positive cycle found in the kernel would render some of its clusters non-maximal -- all the clusters connected by the cycle found should have been grouped together to form a single cluster.

Theorem 2: A network Γ is consistent iff its cycle kernel is locally consistent, i.e., no cycle cluster sprouts a negative self loop or a pair of conflicting arcs pointing to a singleton node.

Proof: The "if" part is proven by repeating the inductive proof of Theorem 1 in any node ordering consistent with the orientation of the positive cross-cluster arcs of Γ_K. For each non-empty set of arcs $S \subseteq E$, we will assign identical values to all nodes in the same cycle cluster of Γ_S. Since the kernel of Γ_S, K_S, is acyclic, the notion of "youngest" Q-node is well defined (translated to "youngest" Q-cluster) and the proof follows through as before with only one modification; whereas Γ_S^+ in Theorem 1 was assumed to contain no conflicting pairs, K_S may contain such pairs under the conditions of Theorem 2, as shown in Figure 4(b). It remains, therefore, to follow the proof of Theorem 1, identify the steps depending on the no-conflicting-pairs assumption, and devise new assignments t, if necessary.

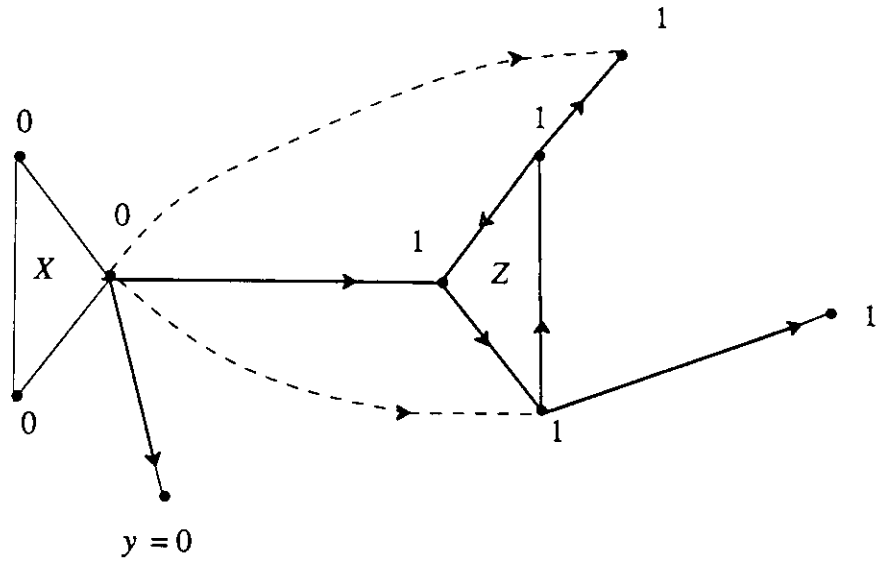

Figure 5

The first use of the no-conflicting-pairs assumption was made in case 1.1 (see Figure 1) where the existence of x' was established. With the assumption removed, a situation like the one described in Figure 5 may prevail, where Z is a cluster receiving both a negative and a positive arc from cluster X. Assigning 1 to Z and all its (positive) descendant, and 0 everywhere else, would confirm S. Note that the existence of x' in the proof of Theorem 1 was essential in case Z was a leaf node of Γ_S^+, because the arc (x', z) is the only one verified by t. Now, since Z is a cluster of at least two arcs, the assignment $Z = 1$ verifies the internal arcs of Z and x' is no longer needed.

The next use of the no-conflicting-pairs assumption was in case 1.2 (see Figure 2). Here we presumed that there is no negative arc pointing from x to y. However, this assumption still hold under Theorem 2 because, even when X is a multi-node cluster, it cannot sprout both a negative and a positive arc toward a singleton y. Moreover, y cannot sprout any positive arc to-

wards any 0-assigned cluster. If it sprouts any positive arcs they must be toward X because the inductive proof is assumed to progress in an order consistent with the orientation of Γ_K^+ so, no positive descendant of y can possibly be in Γ_S unless the two belong to the same cycle cluster.

The no-conflicting-pairs assumption was not used in case 2 because here all positive descendants of x were presumed not to sprout any negative arcs except toward y. This completes the "if" part of the proof.

The "only if" part is proven by showing that, if the local consistency conditions are violated, there exists a non-empty subset of arcs S that cannot be confirmed by any assignment t. First, assume that Γ_K contains a negative self loop. In this case, Γ contains a negative arc $(x, y)^-$ between two nodes belonging to the same cycle-cluster; i.e., there is a sequence of positive arcs $Q = (x, x_1)(x_1, x_2) \cdots (x_r, y)(y, x_{r+1}) \cdots (x_m, x)$ that begins and ends at x and goes through y. Let S be the set of arcs $(x, y)^- \cup \{Q\}$. To verify one arc in S, some vertex in Q should be assigned 1 and, to prevent the falsification of any arc in Q, all vertices should be assigned 1. This falsifies $(x, y)^-$ and, hence, S cannot be confirmed by any assignment t,

A similar argument holds in case Γ_K contains a pair of conflicting arcs, except that we now deal with two arcs; $(x^+, y) \in E^+$ and $(x^-, y) \in E^-$ and both x^+ and x^- are in Q. Let $S = Q \cup (x^+, y) \cup (x^-, y)$ be the subset of arcs requiring confirmation by t. t must assign a value 1 to at least one node in Q, hence all nodes in Q must receive 1 and no assignment for y would keep either (x^+, y) or (x^-, y) from being falsified. Q.E.D.

4. CONCLUSIONS

The importance of deciding consistency lies not only in detecting contradictions in the database, but also in providing a criterion for deciding which sentences are implied by the database, given that it is consistent.

Definition: Given a pair of arbitrary literals x and y and a network Γ, the sentence $x \rightarrow y$ is said to be a *necessary conclusion* of Γ if adding the negative arc $x \rightarrow \neg y$ to Γ would render it inconsistent. For example, since $\Gamma = \{a \rightarrow b, b \rightarrow a, a \rightarrow c, b \rightarrow \neg c\}$ is inconsistent, the negation of any one of these four sentences is a necessary conclusion of the other three.

A necessary conclusion is a rather strong requirement. Very few of the conclusions commonly derived by existing default logics are truly necessary conclusions of the default rules. For example "T flies" is not a necessary conclusion of "T is a bird" and "birds fly" (because T may have a broken wing as an excusable exception). A weaker notion of consistency, involving frame axioms, is required to qualify "T flies" as a conclusion. Interestingly, however, most necessary conclusions of defaults are NOT derivable by existing default logics. For example, "most birds are not penguins" is a necessary conclusion of: birds fly, penguins are birds and penguins do not fly; yet, none of the existing formalisms would generate the first as a conclusion of the latter three.

5. REFERENCES

Adams, E., "Probability and the Logic of Conditionals," in *Aspects of Inductive Logic*, J. Hintikka and P. Suppes (Eds.), North Holland Amsterdam, 1966.

Pearl, J., "Probabilistic Semantics for Inheritance Hierarchies with Exceptions," *TR-93*, UCLA Cognitive Systems Laboratory, July 1987.