Strategies for Determining Causes of Events

Mark Hopkins
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095
mhopkins@cs.ucla.edu

Abstract

In this paper, we study the problem of determining actual causes of events in specific scenarios, based on a definition of actual cause proposed by Halpern and Pearl. To this end, we explore two different search-based approaches, enrich them with admissible pruning techniques and compare them experimentally. We also consider the task of designing algorithms for restricted forms of the problem.

Introduction

Recently, there has been a renewed interest in establishing a precise definition of event-to-event causation, sometimes referred to as "actual cause." Although a definitive answer has been elusive, many proposals have shown promise. This paper focuses on a definition proposed by (Halpern & Pearl 2001), and specifically seeks to find an efficient algorithm to determine whether one event causes another event under this definition.

Complexity results by (Eiter & Lukasiewicz 2001) have shown that in general, the problem of determining actual cause under this definition is Σ_2^P-complete. Therefore, this paper proposes and evaluates search-based strategies, for both the complete and restricted forms of the problem. We are not aware of any other attempts made to address this problem from an algorithmic perspective.

Due to limited space, we provide only proof sketches for some results. Full proofs are available in (Hopkins 2002b).

Formal Description of the Problem

This paper addresses the issue of how to detect whether some event A caused another event B, based on a causal model-based definition proposed in (Halpern & Pearl 2001). Intuitively, the overarching goal is to answer causal queries regarding a fully specified story, and more ambitiously, to generate explanations automatically, in response to "why" questions. An example that illustrates some of the complexities of such a task is called The Desert Traveler, a story inspired by Patrick Suppes and featured in (Pearl 2000).

Example A desert traveler T has two enemies. Enemy 1 poisons T's canteen, and Enemy 2, unaware of Enemy 1's action, shoots and empties the canteen before T drinks. A week later, T is found dead and the two enemies confess to action and intention. A jury must decide whose action was the actual cause of T's death.

Here, the task is to determine whether Enemy 1's action or Enemy 2's action was the actual cause of T's death. (Pearl 2000) phrases the general question in the language of structural causal models.

Before proceeding to formalizations, let us establish some preliminaries. We will generally use upper-case letters (e.g. X, Y) to represent random variables, and the lower-case correspondent (e.g. x, y) to represent a particular value of that variable. $Dom(X)$ will denote the domain of a random variable X. We will use bold-face upper-case letters to represent a set of random variables (e.g. $\mathbf{X}$,$\mathbf{Y}$). The lower-case correspondent (e.g. $\mathbf{x}$,$\mathbf{y}$) will represent a value for the corresponding set. Specifically, for $\mathbf{X} = \{X_1, ..., X_n\}$, a value $\mathbf{x}$ would be a mapping $\mathbf{x} : \mathbf{X} \rightarrow Dom(X_1) \cup ... \cup Dom(X_n)$, such that $\mathbf{x}(X_i) \in Dom(X_i)$. $Dom(\mathbf{X})$ is the set of all possible values for $\mathbf{X}$.

Formally, a *causal model* is a triple $(\mathbf{U}, \mathbf{V}, \mathbf{F})$, in which $\mathbf{U}$ is a finite set of exogenous random variables, $\mathbf{V}$ is a finite set of endogenous random variables (disjoint from $\mathbf{U}$), and $\mathbf{F} = \{F_X | X \in \mathbf{V}\}$ where F_X is a function $Dom(\mathbf{R}) \rightarrow Dom(X)$ that assigns a value to X for each setting of the remaining variables in the model $\mathbf{R} = \mathbf{U} \cup \mathbf{V} \backslash \{X\}$. For each X, we can define $\mathbf{PA}_X$, the *parent set* of X, to be the set of variables in $\mathbf{R}$ that can affect the value of X (i.e. are non-trivial arguments of F_X).

Causal models can be depicted as a *causal network*, a graph whose nodes correspond to the variables in $\mathbf{U} \cup \mathbf{V}$ with an edge from Y to $X \in \mathbf{V}$ if and only if $Y \in \mathbf{PA}_X$. The *endogenous causal network* is defined similarly, but over only the variables in $\mathbf{V}$. *Recursive causal models* are causal models whose causal networks are directed acyclic graphs. In this paper, we restrict our analysis to recursive causal models (following the analysis of (Halpern & Pearl 2001) and (Eiter & Lukasiewicz 2001)).

We will further assume that the domain of each random variable is finite and explicit, that each F_X is computable in polynomial time, and that the number of parents for any particular variable is bounded by a constant.

Example In Fig. 1, we see the Desert Traveler scenario ex-

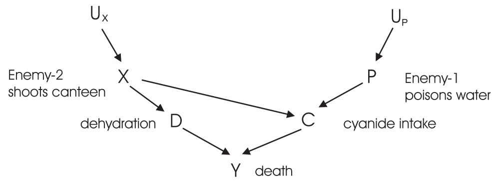

Figure 1: Causal model for the Desert Traveler scenario. All variables are propositional. $X = U_X$; $P = U_P$; $D = X$; $C = P \wedge \neg X$; $Y = C \vee D$.

pressed as a recursive causal model. Here, $\mathbf{U} = \{U_X, U_P\}$ and $\mathbf{V} = \{X, P, D, C, Y\}$. All variables are propositional, with value 1 indicating a true proposition, and value 0 indicating that the proposition is false.

An important aspect of causal models is their ability to store counterfactual information. We can express counterfactual contingencies through the use of submodels. Intuitively, a submodel fixes the values of a set of endogenous variables $\mathbf{X}$ at $\mathbf{x}$. Consequently, the values of the remaining variables represent what values they *would have had* if $\mathbf{X}$ had been $\mathbf{x}$ in the original model. Formally, given a causal model $M = (\mathbf{U}, \mathbf{V}, \mathbf{F})$, $\mathbf{X} \subseteq \mathbf{V}$, $\mathbf{x} \in Dom(\mathbf{X})$, the *submodel* of M under *intervention* $\mathbf{X} = \mathbf{x}$ is $M_{\mathbf{X}=\mathbf{x}} = (\mathbf{U}, \mathbf{V}, \mathbf{F}_{\mathbf{X}=\mathbf{x}})$, where $\mathbf{F}_{\mathbf{X}=\mathbf{x}} = \{F_R | R \in \mathbf{V} \backslash \mathbf{X}\} \cup \{F_{X'} = \mathbf{x}(X') | X' \in \mathbf{X}\}$. $M_{\mathbf{X}=\mathbf{x}}$ and $\mathbf{F}_{\mathbf{X}=\mathbf{x}}$ are typically abbreviated $M_{\mathbf{x}}$ and $\mathbf{F}_{\mathbf{x}}$.

In a recursive causal model, the values of the exogenous variables uniquely determine the values of the endogenous variables. Hence for a causal model $M = (\mathbf{U}, \mathbf{V}, \mathbf{F})$ and a set of endogenous variables $\mathbf{Y} \in \mathbf{V}$ we can refer to the unique value of $\mathbf{Y}$ under $\mathbf{u} \in Dom(\mathbf{U})$ as $\mathbf{Y}_M(\mathbf{u})$ (or simply $\mathbf{Y}(\mathbf{u})$). We can define $\mathbf{Y}_{M_{\mathbf{x}}}(\mathbf{u})$ analogously for a submodel $M_{\mathbf{x}}$ (and abbreviate it $\mathbf{Y}_{\mathbf{x}}(\mathbf{u})$). Since we are dealing with the issue of determining actual cause in a fully-specified scenario, this amounts to asking causal questions in a causal model for which the values of the exogenous variables are given. For causal model $M = (\mathbf{U}, \mathbf{V}, \mathbf{F})$ and $\mathbf{u} \in Dom(\mathbf{U})$, we refer to the pair $(M, \mathbf{u})$ as a *causal world*.

The following properties of recursive causal models, established in (Pearl 2000), will be useful:

Proposition 1 *Let* $M = (\mathbf{U}, \mathbf{V}, \mathbf{F})$ *be a recursive causal model. Let* $\mathbf{u} \in Dom(\mathbf{U})$, $\mathbf{X} \subseteq \mathbf{V}$, $\mathbf{W} \subseteq \mathbf{V}$, $\mathbf{w} \in Dom(\mathbf{W})$, $Y \in \mathbf{V}$. *Then the following properties hold:*

(a) $Y_{\mathbf{w}}(\mathbf{u}) = Y_{\mathbf{wx}}(\mathbf{u})$ *for any* $\mathbf{x} \in Dom(\mathbf{X})$ *if all directed paths from* $\mathbf{X}$ *to* Y *in the causal network of* M *are intercepted by* $\mathbf{W}$.

(b) $Y_{\mathbf{w}}(\mathbf{u}) = Y_{\mathbf{wx}}(\mathbf{u})$ *if* $\mathbf{X}_{\mathbf{w}}(\mathbf{u}) = \mathbf{x}$.

Equipped with this background, we can now proceed to define actual cause:

Definition 2 *Let* $M = (\mathbf{U}, \mathbf{V}, \mathbf{F})$ *be a causal model. Let* $\mathbf{X} \subseteq \mathbf{V}$, $\mathbf{Y} \subseteq \mathbf{V}$. $\mathbf{X} = \mathbf{x}$ *is an* actual cause *of* $\mathbf{Y} = \mathbf{y}$ *(denoted* $\mathbf{x} \propto \mathbf{y}$*) in a causal world* $(M, \mathbf{u})$ *if the following three conditions hold:*

(AC1) $\mathbf{X}(\mathbf{u}) = \mathbf{x}$ *and* $\mathbf{Y}(\mathbf{u}) = \mathbf{y}$.

(AC2) There exists $\mathbf{W} \subseteq \mathbf{V} \backslash \mathbf{X}$ *and values* $\mathbf{x}' \in Dom(\mathbf{X})$ *and* $\mathbf{w} \in Dom(\mathbf{W})$ *such that:*

(a) $\mathbf{Y}_{\mathbf{x}'\mathbf{w}}(\mathbf{u}) \neq \mathbf{y}$.

(b) $\mathbf{Y}_{\mathbf{xw}}(\mathbf{u}) = \mathbf{y}$.

(c) $\mathbf{Y}_{\mathbf{xwz}}(\mathbf{u}) = \mathbf{y}$, *for all* $\mathbf{Z} \subseteq \mathbf{V} \backslash (\mathbf{X} \cup \mathbf{W})$ *such that* $\mathbf{z} = \mathbf{Z}(\mathbf{u})$.

(AC3) $\mathbf{X}$ *is minimal; no subset of* $\mathbf{X}$ *satisfies conditions AC1 and AC2.*

Intuitively, $\mathbf{x}$ is an actual cause of $\mathbf{y}$ if (AC1) $\mathbf{x}$ and $\mathbf{y}$ are the "actual values" of $\mathbf{X}$ and $\mathbf{Y}$ (i.e. the values of $\mathbf{X}$ and $\mathbf{Y}$ under no intervention), and (AC2) under some counterfactual contingency $\mathbf{w}$, the value of $\mathbf{Y}$ is dependent on $\mathbf{X}$, such that setting $\mathbf{X}$ to its actual value will ensure that $\mathbf{Y}$ maintains its "actual value," even if we force all other variables in the model back to their "actual values." (AC3) is a simple minimality condition.

Example In the Desert Traveler example, we see that $X = 1$ (shooting the canteen) is indeed an actual cause of $Y = 1$ (death), since $X(\mathbf{u}) = 1$, $Y(\mathbf{u}) = 1$, $Y_{X=0,C=0}(\mathbf{u}) = 0$, and $Y_{X=1,C=0}(\mathbf{u}) = 1$. Here, our $\mathbf{w}$ is $C = 0$. Notice also that $P = 1$ is *not* an actual cause of $Y = 1$ under this definition.

The question that this paper addresses is: given a causal world $(M, \mathbf{u})$, how can we efficiently determine whether $\mathbf{x}$ is an actual cause of $\mathbf{y}$? Unfortunately, it turns out that this problem is Σ_2^P-complete (Eiter & Lukasiewicz 2001).

Because of this, we focus on search strategies for determining actual cause. For simplicity, we will be restricting our examination to single variable causation, i.e. whether $X = x$ causes $Y = y$, for $X, Y \in \mathbf{V}$. This restriction is partially justified by the following theorem, proven in (Eiter & Lukasiewicz 2001), and independently in (Hopkins 2002a):

Theorem 3 *Let* $M = (\mathbf{U}, \mathbf{V}, \mathbf{F})$ *be a causal model. Let* $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}$ *and* $\mathbf{x} \in Dom(\mathbf{X})$, $\mathbf{y} \in Dom(\mathbf{Y})$. *If* $\mathbf{x} \propto \mathbf{y}$ *under* $\mathbf{u}$, *then* $\mathbf{X}$ *is a singleton.*

This theorem establishes that any candidate cause that contains multiple variables will inevitably violate the minimality requirement of the actual cause definition. Thus we may restrict our focus to singleton causes.

We also do not consider our effect to be a Boolean conjunction of primitive events $\mathbf{Y} = \mathbf{y}$, since $X = x$ is an actual cause of $(\mathbf{Y} = \mathbf{y} \wedge \mathbf{Z} = \mathbf{z})$ iff $X = x$ is an actual cause of $\mathbf{Y} = \mathbf{y}$ and $X = x$ is an actual cause of $\mathbf{Z} = \mathbf{z}$. Thus any algorithm that determines actual cause between primitive events can immediately be applied in the more general case through repeated applications of the algorithm.

Thus, let us consider the task of determining whether $x \propto y$ holds in a given causal world. The first thing to notice is that checking AC1 and AC3 are easy tasks. To check AC1, we merely need to check the value of two different random variables under a single intervention (specifically, the null intervention). We can compute the value of every random variable in a causal world under a single intervention in polynomial time, by the following simple procedure: choose a variable for whom the values of the parents are determined, then compute the value for that variable; continue until values for all variables are computed. Clearly, this procedure is always executable in a recursive causal world. AC3 is trivial, in light of Thm. 3.

The difficulty lies in checking whether or not AC2 holds. The remainder of this paper deals with strategies for deciding this. We should point out that the task of determining whether AC2 holds boils down to searches through two different search spaces:

1. A search through possible $\mathbf{w}$. The top-level task is to find a set of variables $\mathbf{W}$, and a particular value $\mathbf{w} \in Dom(\mathbf{W})$ that satisfies all three constraints of AC2.
2. A search through possible $\mathbf{Z}$. Notice that for a given $\mathbf{w}$, AC2(a) and AC2(b) can be checked in polynomial time (since each merely requires us to compute the value of a variable under a single intervention). However, AC2(c) is more involved. It requires us to check that there is no set $\mathbf{Z}$ such that $Y_{\mathbf{xwz}}(\mathbf{u}) \neq y$, where $\mathbf{z} = \mathbf{Z}(\mathbf{u})$. Here we are searching for a set of variables, rather than for a particular value for a set of variables, as in the search for $\mathbf{w}$.

Algorithm-Independent Optimizations

Naturally we would like to reduce the size of these search spaces as much as possible. To this end, we define the notion of the *projection* of a causal world.

Definition 4 *Let $M = (\mathbf{U}, \mathbf{V}, \mathbf{F})$ be a recursive causal model. Suppose we have a causal world $(M, \mathbf{u})$ such that $\mathbf{V} = \{V_1, ..., V_n\}$. To* delete *a variable V_i from $(M, \mathbf{u})$, V_i is removed from $\mathbf{V}$, and the structural equation F_X of each child X of V_i is replaced with $F_X|_{v_i}$, where $v_i = V_i(\mathbf{u})$. The* projection *of $(M, \mathbf{u})$ over variables $V_1, V_2, ..., V_k$ is a new causal model in which $V_{k+1}, V_{k+2}, ..., V_n$ are deleted from $(M, \mathbf{u})$. The* W-projection *of $(M, \mathbf{u})$ with respect to $x \propto y$ is the projection of $(M, \mathbf{u})$ over X, Y, the variables $\mathbf{V}^{XY}$ on a path from X to Y in the causal network of M, and the parents of $\mathbf{V}^{XY}$ and Y in the causal network of M.*

Intuitively, *deleting* a variable gives us the same result as permanently fixing it at its actual value. Now we can prove that the question of whether x is an actual cause of y in $(M, \mathbf{u})$ depends only on the paths that connect X to Y in the causal network of M, and the nodes which influence nodes on these paths (i.e. the W-projection). All other nodes either do not influence Y, or do so through a parent of a node on a path, and can be safely ignored.

Theorem 5 *Let $M = (\mathbf{U}, \mathbf{V}, \mathbf{F})$ be a recursive causal model and suppose we have a causal world $(M, \mathbf{u})$. Then $x \propto y$ in $(M, \mathbf{u})$ iff $x \propto y$ in $(M', \mathbf{u})$, where M' is the W-projection of $(M, \mathbf{u})$.*

Proof Suppose $x \propto y$ in $(M, \mathbf{u})$. Then we must have $\mathbf{W} \subseteq \mathbf{V} \backslash X$, $\mathbf{w} \in Dom(\mathbf{W})$, and $x' \in Dom(X)$ such that AC2 is satisfied. Now consider the set $\mathbf{P}$ of variables which are parents to variables on a path from X to Y (except parents of X), but are not themselves on a path from X to Y. Suppose $\mathbf{P_w}(\mathbf{u}) = \mathbf{p}$. Define $\mathbf{W}'$ as the union of $\mathbf{P}$ with the subset of $\mathbf{W}$ on a directed path from X to Y, and define $\mathbf{w}'$ such that $\mathbf{w}'(V) = \mathbf{w}(V)$ for $V \in \mathbf{W}$ and $\mathbf{w}'(V) = V_{\mathbf{w}}(\mathbf{u})$ for $V \in \mathbf{P}$.

We will show that this $\mathbf{w}' \in Dom(\mathbf{W}')$ also satisfies AC2. By Prop. 1(a), $Y_{x'\mathbf{w}'}(\mathbf{u}) = Y_{x'\mathbf{pw}}(\mathbf{u})$. Then, since $P_{\mathbf{w}}(\mathbf{u}) = P_{x'\mathbf{w}}(\mathbf{u})$ by Prop. 1(a), therefore $Y_{x'\mathbf{pw}}(\mathbf{u}) = Y_{x'\mathbf{w}}(\mathbf{u})$ by Prop. 1(b). Hence $Y_{x'\mathbf{w}'}(\mathbf{u}) = Y_{x'\mathbf{w}}(\mathbf{u}) \neq y$, so $\mathbf{w}'$ satisfies AC2(a).

Now we show that it satisfies AC2(b) and (c). Take any $\mathbf{Z} \subseteq \mathbf{V} \backslash (X \cup \mathbf{W})$ and let $\mathbf{z} = \mathbf{Z}(\mathbf{u})$. By Prop. 1(a), $Y_{x\mathbf{w}'\mathbf{z}}(\mathbf{u}) = Y_{x\mathbf{pwz}}(\mathbf{u})$. Also by Prop. 1(a), $Y_{x\mathbf{pwz}}(\mathbf{u}) = Y_{x\mathbf{pwz}'}(\mathbf{u})$, where $\mathbf{Z}'$ is the subset of $\mathbf{Z}$ on a directed path from X to Y, and $\mathbf{z}' = \mathbf{Z}'(\mathbf{u})$. Since $P_{\mathbf{w}}(\mathbf{u}) = P_{x\mathbf{wz}'}(\mathbf{u})$ by Prop. 1(a), therefore $Y_{x\mathbf{pwz}'}(\mathbf{u}) = Y_{x\mathbf{wz}'}(\mathbf{u})$ by Prop. 1(b). Hence $Y_{x\mathbf{w}'\mathbf{z}}(\mathbf{u}) = Y_{x\mathbf{wz}'}(\mathbf{u}) = y$, so $\mathbf{w}'$ satisfies AC2(b) and AC2(c).

Hence we can always devise an intervention consisting only of variables on a path from X to Y and variables in $\mathbf{P}$ that satisfies AC2. Clearly, AC1 and AC3 are also satisfied in $(M', \mathbf{u})$. Hence, $X = x$ is an actual cause of $Y = y$ in $(M', \mathbf{u})$. The converse of this theorem is trivial. ∎

Looking at Fig. 2, we see how this can substantially reduce the number of nodes in the causal network (while not increasing the connectivity). Notice that such projection can be done before attempting *any algorithm* for determining whether x is an actual cause of y. Henceforth, unless stated otherwise, we will implicitly assume that our algorithms are operating on the W-projection with respect to the query.

Brute-Force Approach

Recall that the task of determining whether AC2 holds can be divided into two stages: a search through possible $\mathbf{w}$, and a search through possible $\mathbf{Z}$. For the next two sections, we will focus on the first stage, and assume that we have a black box to check AC2(c).

There is a rather obvious brute-force search algorithm through the space of possible $\mathbf{w}$. For some ordering of the variables of our causal world (excluding Y), we simply assign a value (including a possible "non-assignment" value $\emptyset$, indicating that the variable is not part of $\mathbf{W}$) to each variable, one at a time, until all variables have been assigned values. Then we check to see whether AC2 is satisfied by this $\mathbf{w}$. Here, we are lumping X into $\mathbf{W}$, thus to check AC2(a) and (b), we simply check that $Y_{\mathbf{w}}(\mathbf{u}) \neq y$ and $Y_{\mathbf{w}'x}(\mathbf{u}) = y$, where $\mathbf{w}'$ is merely $\mathbf{w}$ with the setting for X removed. If AC2(a) and (b) hold, then we check AC2(c) with our black box. If this also holds, then $x \propto y$. If not, we try another setting of the variables, until we find one that satisfies AC2 or until all possibilities are exhausted.

The obvious drawback of this approach is that it amounts to a brute-force search of a tree with depth $N-1$ and branching factor $c+1$, where N is the number of variables in the

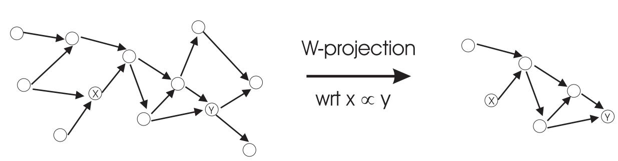

Figure 2: Demonstration of W-Projection

causal network (minus one, for Y), and c is the size of each variable domain in the network (plus one, for $\emptyset$) Here, we assume for simplicity that each variable domain has the same size. Thus this search tree has $O((c+1)^N)$ leaf nodes.

To check whether any given leaf in the search tree satisfies AC2(a) and (b), we potentially need to calculate the value of each node in our network under two different interventions – as we have discussed, this task is polynomial in the number of endogenous variables. Supposing now that $O(\lambda)$ is the worst-case complexity of the algorithm to check AC2(c), we can say that the worst-case running time of this brute-force algorithm is $O(\lambda N^p(c+1)^N)$, for some constant p.

One approach to pruning the brute-force search tree is based on the following theorem:

Theorem 6 *Let N be the causal network of recursive causal model $M = (\mathbf{U}, \mathbf{V}, \mathbf{F})$. Let $\mathbf{u} \in Dom(\mathbf{U})$, $X, Y \in \mathbf{V}$, $x, x' \in Dom(X)$, $y \in Dom(Y)$. Suppose $\mathbf{W} \subseteq \mathbf{V}\backslash X$ and that every path from X to Y in N is blocked by some variable in $\mathbf{W}$. Then for any $\mathbf{w} \in Dom(\mathbf{W})$, $Y_{x'\mathbf{w}}(\mathbf{u}) = Y_{x\mathbf{w}}(\mathbf{u})$. In other words, either AC2(a) or AC2(b) must fail. Furthermore, any $\mathbf{W}' \supseteq \mathbf{W}$ also has this property.*

Proof By Prop. 1(a), $Y_{x'\mathbf{w}'}(\mathbf{u}) = Y_{\mathbf{w}'}(\mathbf{u}) = Y_{x\mathbf{w}'}(\mathbf{u})$, for any $\mathbf{W}' \supseteq \mathbf{W}$, $\mathbf{w}' \in Dom(\mathbf{W}')$. ■

Theorem 6 implies that we can prune any subtree rooted at node S from our search tree, where S is any node representing a partial variable setting I for which every path from X to Y in the causal network contains some variable set by I to a non- $\emptyset$ value.

The cost per node of determining whether all paths from X to Y are blocked is simply the cost of a depth-first search of the graph, which is linear in the number of network nodes. Thus adding this pruning to our brute-force algorithm does not add any complexity to our asymptotic running time.

The benefit of this pruning will depend on the order in which we assign values to variables, and this paper does not address this issue.

Intervention-Proving Approach

Let us continue to treat AC2(c) as a black box, and examine a different approach to searching through the space of possible $\mathbf{w}$. To determine whether AC2(a) and (b) are satisfied for query $x \propto y$, we are looking for some $\mathbf{w}$, such that $Y_{x\mathbf{w}}(\mathbf{u}) = y$ and $Y_{x'\mathbf{w}}(\mathbf{u}) \neq y$ for some $x' \neq x$. We can actually encode part of this goal into the search space itself. For example, we could choose to only search the space of interventions $\mathbf{w}'$ such that $Y_{\mathbf{w}'}(\mathbf{u}) \neq y$. Then for each intervention of the form $\mathbf{w}' = x'\mathbf{w}$, we would need only to check that $Y_{x\mathbf{w}} = y$.

We can think of this as "proving" interventions $\mathbf{w}'$ such that $Y_{\mathbf{w}'}(\mathbf{u}) \neq y$ in the causal model. We begin by supposing that $Y \neq y$, and work backwards to prove which interventions would be consistent with $Y \neq y$. Take, for instance, the Desert Traveler formulation. Suppose we want to determine whether $X = 1$ is an actual cause of $Y = 1$. Suppose that $Y \neq 1$. What are the possible instantiations of Y's parents in that case? In fact, the only possibility is $\{D = 0, C = 0\}$. It is easy to see that under every intervention $\mathbf{w}'$ such that $Y_{\mathbf{w}'}(\mathbf{u}) \neq 1$, it must be the case that $D_{\mathbf{w}'}(\mathbf{u}) = 0$ and $C_{\mathbf{w}'}(\mathbf{u}) = 0$. More importantly, our claim is that $Y_{D=0,C=0}(\mathbf{u}) \neq 1$. Now let us take this new intervention we have produced, and attempt to eliminate C from it in the same way. Notice that the intervention requires that condition $C = 0$ will materialize. There are three full instantiations of the parents of C that force C's value to be 0: $\{X = 0, P = 0\}$, $\{X = 1, P = 0\}$, $\{X = 1, P = 1\}$. Thus, we can replace $C = 0$ with these three instantiations to create three new interventions. Furthermore, if we continue this process (eliminating the variables in a reverse topological order), as shown in Fig. 3, then for every intervention $\mathbf{w}'$ we produce, $Y_{\mathbf{w}'}(\mathbf{u}) \neq 1$.

A few observations are in order. At certain points, it is possible to derive inconsistent interventions. These are immediately thrown out (marked by X's in the figure). Also, each level of the tree can be thought of as "eliminating" a single variable in the manner described above. However, at each level, we may also choose *not* to eliminate the variable (indicated by the rightmost child of each node).

Once we have derived a number of interventions $\mathbf{w}'$ such that $Y_{\mathbf{w}'}(\mathbf{u}) \neq y$, we simply take the ones of the form $\mathbf{w}' = x'\mathbf{w}$, and check to see if $Y_{x\mathbf{w}} = y$ and AC2(c) are satisfied. The relevant interventions are circled in the figure. Notice that the intervention $\{X = 0, C = 0\}$ is discovered.

The pseudocode for the general intervention-proving algorithm is presented in Figure 4. Notice that Figure 3 does not correspond exactly to the tree searched by Figure 4, in the sense that if we choose to keep the setting for variable $I(k)$ at depth k of the tree, we do not actually generate a duplicate intervention at depth $k + 1$. Instead, we merely keep track of our "virtual depth" in the tree that ensures that we only eliminate variables in reverse topological order.

It is clear that if the intervention-proving search tree generates every intervention $\mathbf{w}'$ such that $Y_{\mathbf{w}'}(\mathbf{u}) \neq y$, and no intervention $\mathbf{w}'$ such that $Y_{\mathbf{w}'}(\mathbf{u}) = y$, then this algorithm will work correctly. Indeed, we can prove half of this.

Lemma 7 *For every intervention $\mathbf{w}'$ found by $IP((M, \mathbf{u}), x, y)$, $Y_{\mathbf{w}'}(\mathbf{u}) \neq y$.*

Proof sketch For a node N in the IP search tree, define $\mathbf{w}(N)$ as the intervention that N represents. It can be proven by induction that if $\mathbf{w}(N)$ sets variable V to v, then for any descendant D of N in the IP search tree, we have $V_{\mathbf{w}(D)} = v$. Thus, since the root of the IP search tree generates only

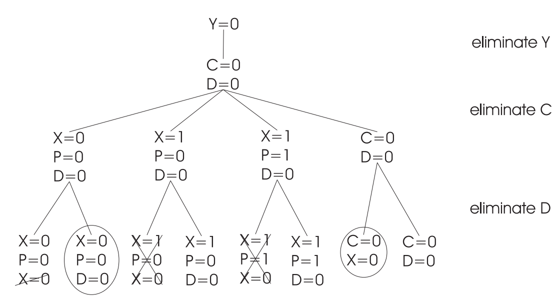

Figure 3: Intervention-Proving Search Tree for the Desert Traveler Scenario (query: $(X = 1) \propto (Y = 1)$)

```
Algorithm IP(CausalWorld (M,u), Cause X=x, Effect Y=y):
1. Let M' be the W-projection of (M,u) wrt x ∝ y.
2. Let I(n) be a reverse topological order of the
   endogenous variables of M' such that Y=I(0).
3. For all interventions y' such that y'≠ y:
   - if(IPTreeWalk(y', 0) == true) then return true.
4. Return false.

bool IPTreeWalk( Intervention w', int treedepth):
1. If w' is of the form x'w, then check that Y_xw(u) = y.
   If so, then check that w satisfies AC2(c). If so, then
   return true.
2. For every (single) variable assignment V=v in w'
   such that V ≠X, V=I(k) for k >= treedepth:
   3. For every full instantiation p of V's parents such
      that F_V(p) = v:
      4. w2 ← (w' \ v) ∪ p.
      5. If w2 is internally consistent, then return true
         if IPTreeWalk(w2, k) returns true
6. Return false.
```

Figure 4: Pseudocode for the IP Algorithm

nodes corresponding to interventions of the form $Y = y'$ such that $y' \neq y$, Lem. 7 follows as a simple corollary. ∎

Although it is not the case that IP generates *every* intervention $\mathbf{w}'$ such that $Y_{\mathbf{w}'}(\mathbf{u}) \neq y$, we can nevertheless characterize the interventions that it produces.

Lemma 8 *$IP((M,\mathbf{u}),x,y)$ finds only interventions $\mathbf{w}'$ subject to the following two conditions:*
(a) $\mathbf{W}'$ contains a node on every path from a root node to Y in the endogenous causal network of M
(b) If $\mathbf{W}' \backslash \{V\}$ contains a node on every path from variable V to Y, then $V \notin \mathbf{W}'$.

Proof sketch The proof is a straightforward induction on the depth of the IP search tree. ∎

Lemma 9 *$IP((M,\mathbf{u}),x,y)$ finds every intervention $\mathbf{w}'$ such that $Y_{\mathbf{w}'}(\mathbf{u}) \neq y$, subject to conditions (a) and (b) of Lem. 8.*

Proof sketch For node N in the IP search tree, define $\mathbf{w}(N)$ as the intervention that N represents. Define $\mathbf{W}(N)$ as the set of endogenous variables fixed by $\mathbf{w}(N)$. Let $\mathbf{w}'$ be an intervention satisfying the conditions of the lemma.

Let $Q(k)$ be the following proposition: "At depth k of the IP search tree, there exists a node N such that for all $V \in \{I(j)|j <= k-1\}$, we have: (i) $V \notin \mathbf{W}' \Leftrightarrow V \notin \mathbf{W}(N)$, (ii) $V \in \mathbf{W}' \Leftrightarrow V \in \mathbf{W}(N)$, (iii) $\forall V \in \mathbf{W}(N), V_{\mathbf{w}'}(\mathbf{u}) \in \mathbf{w}(N)$." $Q(k)$ can be proven by induction on k. From this result, the lemma immediately follows, since it implies that at depth $n+1$ of the IP search tree, IP will find a node N such that $\mathbf{w}(N) = \mathbf{w}'$. ∎

From these lemmas, we can prove the correctness of IP.

Theorem 10 *IP returns as output "$x \propto y$" iff $x \propto y$.*

Proof From Lem. 7, it is clear that IP returns "$x \propto y$" only if $x \propto y$, since IP only finds interventions that satisfy AC2(a), then checks to see if they satisfy AC2(b) and (c).

Now suppose that $x \propto y$. Then there is some intervention $x'\mathbf{w}$ that satisfies AC2. Now consider the set of root nodes $\mathbf{R} = \{R_i | R_i \notin \mathbf{W} \cup \{X\}\}$ in the endogenous causal network. Create a new intervention $x'\mathbf{wr}$, where $\mathbf{r} = \mathbf{R}(\mathbf{u})$. Notice that since $\mathbf{R}$ are root nodes, $\mathbf{R}(\mathbf{u}) = \mathbf{R}_{x'\mathbf{w}}(\mathbf{u})$, thus this new intervention must also satisfy AC2. Now consider any node $S \in \mathbf{W} \cup \mathbf{A}$ for which every path from S to Y contains some node in $\mathbf{W} \backslash \{S\}$. Notice that the value of Y is therefore independent of the value of S, and hence we can remove S from our intervention and this new intervention will still satisfy AC2. Thus we can always create a new intervention that satisfies AC2 and also satisfies the two conditions specified in Lem. 8. Hence by Lem. 9, IP will find this intervention, and return "$x \propto y$". ∎

The running time of this algorithm will vary substantially, depending on the topology and quantification of the causal network, but we can say with certainty that in the worst-case, IP generates no more nodes than the brute-force search tree of the previous section.

Theorem 11 *The search tree generated by IP contains no duplicate nodes, i.e. no two nodes that represent the same intervention.*

Proof sketch Proof by contradiction. Assume that there are two distinct nodes D and E in the IP search tree that represent the same intervention. Let A be the common ancestor of D and E, and let B and C be the children of A on the path to D and E, respectively. Then it can be shown that B and C must differ on the value of at least one variable set in their respective interventions, and that this difference propagates down to D and E. Hence D and E must represent different interventions. ∎

Since the tree generated by IP contains no duplicate nodes, IP generates asympotOtotically no more nodes than the brute-force search tree, since that tree contains nodes representing every possible intervention, and the IP search tree contains only a subset of possible interventions (note that in addition, IP generates a certain number of inconsistent nodes, then immediately throws them away; this adds only a constant amount of work per node, and thus does not impact asymptotic running time). We can characterize the worst-case number of nodes that IP will generate in terms of the size of the subset of interventions that it generates. Define S as the subset of interventions $\mathbf{w}'$ such that $Y_{\mathbf{w}'}(\mathbf{u}) \neq y$ and $\mathbf{w}'$ satisfies the conditions specified in Lem. 8. Then from Lem. 8, IP generates $|S|$ nodes in the worst-case.

For each node, we need to check that AC2(b) holds, which as we have seen, is polynomial in the number of nodes N in the causal network. We also need to examine every table entry for the variable we are eliminating, in order to generate the children of the node. This takes $O(c^k)$ time, where c is the maximum size of the domain of the network variables and k is the maximum number of parents per node. We will assume that the number of parents per node is bounded by a constant, as is the maximum size of the domain of the network variables. Hence c^k becomes a constant. Once again supposing that checking AC2(c) takes $O(\lambda)$ time, we can say that the worst-case asymptotic running time of IP is $O(c^k \lambda N^p |S|) = O(\lambda N^p |S|)$ Borrowing from the results we derived in the previous section, we can say that $O(\lambda N^p|S|) = O(\lambda N^p(c+1)^N)$. It is not clear how much tighter a bound $O(\lambda N^p|S|)$ is, compared with $O(\lambda N^p(c+1)^N)$. Experimental results, however, suggest that the difference is quite significant.

Checking AC2(c)

Once we find an intervention $\mathbf{w}$ that satisfies AC2(a) and AC2(b), we then face the challenge of checking whether AC2(c) is also satisfied by $\mathbf{w}$. To do this, we need to search through the space of possible $\mathbf{Z}$. For each $\mathbf{Z}$, we need to check that $Y_{x\mathbf{wz}} = y$, where $\mathbf{z} = \mathbf{Z}(\mathbf{u})$.

For a particular $\mathbf{Z}$, it is not difficult to check this. We merely need to compute the value of Y under an intervention, which can be done in polynomial time. The problem is that if $\mathbf{Z}^C$ represents the set of variables that are candidates for inclusion in $\mathbf{Z}$, then there are $2^{|\mathbf{Z}^C|}$ possible $\mathbf{Z}$.

Thus, one critical issue is to limit the size of $\mathbf{Z}^C$, the candidate set for $\mathbf{Z}$. Unfortunately, the definition itself specifies that $\mathbf{Z}^C$ contains every variable in the causal world that is not X, Y, or a member of $\mathbf{W}$. Fortunately, we can do better.

Theorem 12 *Define $x \propto' y$ similarly to $x \propto y$, except with AC2(c) replaced by the following:*

AC2(c)′: $Y_{xwz}(\mathbf{u}) = y$, for all $\mathbf{Z} \subseteq \mathbf{Z}^C$ such that $\mathbf{z} = \mathbf{Z}(\mathbf{u})$ and $\mathbf{Z}^C = \{V \in \mathbf{V}\backslash(X \cup \mathbf{W}) | V$ appears on one or more directed paths from X to Y in the causal network of M that do not contain a member of $\mathbf{W}\}$.

Then $x \propto' y$ iff $x \propto y$.

Proof Suppose $x \propto' y$. Then there exists some intervention $x'\mathbf{w}$ that satisfies the modified version of AC2. Consider the set $\mathbf{Z}^A = (\mathbf{V}\backslash(\{X\} \cup \mathbf{W}))\backslash\mathbf{Z}^C$. For all $Z \in \mathbf{Z}^A$, either $\mathbf{W}$ intercepts all paths from Z to Y in the endogenous causal network of M, or $\mathbf{W}$ intercepts all paths from X to Z (otherwise $Z \in \mathbf{Z}^C$). Thus define $\mathbf{Z}^{XZ}$ as $\{Z \in \mathbf{Z}^A | \mathbf{W}$ intercepts all paths from X to $Z\}$, and $\mathbf{Z}^{ZY}$ as $\mathbf{Z}^A\backslash\mathbf{Z}^{XZ}$. Create a new intervention $x'\mathbf{w}\mathbf{Z}^{XZ}_{x'\mathbf{w}}(\mathbf{u})$. We want to show that this new intervention satisfies the original version of AC2.

Since by Prop. 1(b) $Y_{x'\mathbf{w}\mathbf{Z}^{XZ}_{x'\mathbf{w}}(\mathbf{u})}(\mathbf{u}) = Y_{x'\mathbf{w}}(\mathbf{u}) \neq y$, clearly AC2(a) is satisfied. Using Prop. 1(a) and (b), $Y_{x\mathbf{w}\mathbf{Z}^{XZ}_{x'\mathbf{w}}(\mathbf{u})}(\mathbf{u}) = Y_{x\mathbf{w}\mathbf{Z}^{XZ}_{x\mathbf{w}}(\mathbf{u})}(\mathbf{u}) = Y_{x\mathbf{w}}(\mathbf{u}) = y$, so AC2(b) is also satisfied. Finally, take any $\mathbf{Z} \subseteq (\mathbf{V}\backslash(\{X\} \cup \mathbf{W} \cup \mathbf{Z}^{XZ}))$. $Y_{x\mathbf{w}\mathbf{Z}^{XZ}_{x'\mathbf{w}}(\mathbf{u})\mathbf{Z}(\mathbf{u})}(\mathbf{u}) = Y_{x\mathbf{w}\mathbf{Z}^{XZ}_{x'\mathbf{w}}(\mathbf{u})\mathbf{Z}'(\mathbf{u})}(\mathbf{u})$, where $\mathbf{Z}'$ is $\mathbf{Z}$ with all members of $\mathbf{Z}^{ZY}$ removed (by Prop. 1(a), since $\mathbf{W}$ intercepts all paths from these variables to Y). Furthermore, using Prop. 1(a) and (b), $Y_{x\mathbf{w}\mathbf{Z}^{XZ}_{x'\mathbf{w}}(\mathbf{u})\mathbf{Z}'(\mathbf{u})}(\mathbf{u}) = Y_{x\mathbf{w}\mathbf{Z}^{XZ}_{x\mathbf{w}}(\mathbf{u})\mathbf{Z}'(\mathbf{u})}(\mathbf{u}) = Y_{x\mathbf{w}\mathbf{Z}'(\mathbf{u})}(\mathbf{u}) = y$, so AC2(c) is satisfied.

Hence $x \propto' y \rightarrow x \propto y$. The converse is trivial. ∎

Given this result, we can construct a binary search tree to check AC2(c) in the following manner: for a given ordering of the variables in $\mathbf{Z}^C$, assign each variable to either be included in $\mathbf{Z}$ or to not be included in $\mathbf{Z}$. The leaves of such a tree will then be the possible $\mathbf{Z}$ we need to check. Hence, we can apply a simple dfs, and whenever we hit a leaf, check that the $\mathbf{Z}$ represented by the leaf is consistent with AC2(c). If not, then AC2(c) fails. If all leaves are consistent with AC2(c), then AC2(c) holds.

We can define the *value* of this search tree as 1 if all leaves are consistent with AC2(c), and 0 otherwise. The search tree can be pruned, by use of the following theorem:

Theorem 13 *Let N be a node of this search tree. N represents the choice of a certain subset $\mathbf{Z}(N)$ of $\mathbf{Z}^C$ for inclusion in $\mathbf{Z}$. If there exists a variable $V \in \mathbf{Z}^*$ such that every path from V to Y in the causal network is blocked by some other variable in $\mathbf{Z}(N)$, then the subtree rooted at this node can be pruned from the search tree with no change to the value of the tree.*

Proof Suppose that there is some $\mathbf{Z}$ such that $\mathbf{Z}(N) \subseteq \mathbf{Z}$ and such that $Y_{x\mathbf{w}\mathbf{Z}(\mathbf{u})}(\mathbf{u}) \neq y$. If we have variable $V \in \mathbf{Z}(N)$ for which every path from V to Y is blocked by some other variable in $\mathbf{Z}(N)$, then by Prop. 1(a), $Y_{x\mathbf{w}\mathbf{Z}(\mathbf{u})}(\mathbf{u}) = Y_{x\mathbf{w}(\mathbf{Z}\backslash\{V\})(\mathbf{u})}(\mathbf{u})$. Hence there exists a $\mathbf{Z}'$ in another subtree of the search tree that also violates AC2(c). Thus we can prune the subtree rooted at N with no change to the overall value of the tree. ∎

This paper does not address which variable ordering heuristics can help to maximize the impact of such pruning.

Restricted Forms

So far, we have outlined only complete strategies for handling the general problem of determining $x \propto y$. In this section, we consider whether we can develop better algorithms for restricted forms of the problem.

(Nebel 1996) states that intuitively speaking, a problem in Σ_2^P suggests two sources of complexity. We have identified these sources as the search for $\mathbf{w}$ and the search for $\mathbf{Z}$. In order to achieve a polynomial-time algorithm for actual cause, we would need to eliminate both sources of complexity. Unfortunately, to do so, we would likely be restricting the problem to such an extent as to render the solution useless in practice. Nevertheless, we can try to eliminate one of the sources of complexity to improve the speed of our algorithm (although the algorithm will still be exponential-time).

One method of doing so takes advantage of the following result from (Eiter & Lukasiewicz 2001):

Theorem 14 *Let M be a causal world for which all variables are binary. Suppose for a given $\mathbf{x}, \mathbf{y}, \mathbf{w}$, AC1, AC2(a), and AC2(b) hold. Then AC2(c) holds iff $\mathbf{Y}_{\mathbf{xwz}}(\mathbf{u}) = \mathbf{y}$, where $\mathbf{z} = \mathbf{Z}(\mathbf{u})$ and $\mathbf{Z} = \mathbf{V}\backslash(\mathbf{X} \cup \mathbf{W})$.*

In other words, under a binary causal world, there is no need to search through the space of possible $\mathbf{Z}$. It is sufficient to simply check the set $\mathbf{V}\backslash(\{X\} \cup \mathbf{W})$. This amounts to checking the value of Y under a single intervention, which as we have noted, takes polynomial time. Thus we can replace our exponential-time AC2(c) check with a simple polynomial-time check. Hence the asymptotic running time of IP becomes $O(N^p|S|)$, whereas the more general procedure requires $O(2^{|\mathbf{Z}^C|}N^p|S|)$.

Experimental Results

To test the algorithms outlined in this paper, we generated random causal worlds through the following process:

1. We generated a random DAG over n variables by adding an edge from variable k to variable l, $k < l$ with probability P_E. We also limited the number of parents allowed per node at L.
2. We quantified the table for variable V by randomly choosing the value of each table entry from a uniform distribution over the domain (of size D) of V.

Let V_1 be variable 1, and let $v_1 = V_1(\mathbf{u})$. Let V_n be variable n, and let $v_n = V_n(\mathbf{u})$. The query to our algorithms was $v_1 \propto v_n$. Note that V_1 is a root of the endogenous causal network, and V_n is a leaf.

We first tested the average size of the W-projection of a randomly generated causal world. We generated 2000 random networks by the process described above (with $L = 3$, $D = 2$), then pruned each with regard to $v_1 \propto v_n$. The averaged results are presented in Table 1. Such pruning can provide dramatic results for lower values of P_E.

We then implemented three algorithms: the brute-force algorithm, the same algorithm with the tree pruning described by Thm. 6, and the IP algorithm. Each used the CheckAC2c procedure with the pruning described by Thm. 13. For the brute-force algorithm with pruning, we used an arbitrary topological order of the causal network variables as our variable ordering. To compare these algorithms, we generated 5000 random causal worlds over 25 variables by the process described above with $P_E = .15, L = 3$, and $D = 3$. Then we computed the W-projection of each world with respect to query $x \propto y$. Finally, we ran each algorithm on the W-projections (on a Sun Ultra 10 workstation). The results are presented in Table 2, where N is the number of variables in the W-projection (hence $N \leq 25$). We display only values of N from 2 to 18. Clearly, IP enjoys a considerable advantage over the brute-force approach with pruning. Observe that the average time to generate each node seems to be larger for IP than for the brute-force algorithms (by a factor of about 2 or 3). Still, the savings that IP provides in terms of the total number of generated nodes easily makes up for this cost. Moreover, the performance of IP on binary worlds shows an even greater contrast, with mean execution time of 40 seconds and 20000 generated nodes on 18-node W-projections.

Conclusions

In this paper, we have presented basic algorithms for determining actual cause according to the definition presented in (Halpern & Pearl 2001). First, we presented a method of reducing the problem size by projecting a causal world onto a reduced set of (query-dependent) variables. Then, we explored two approaches to solving the problem and devised proven methods of pruning the search space. The second attempt, the intervention-proving approach, achieved superior experimental results. Finally, we considered the task of deriving algorithms for restricted forms of the problem, and showed how the IP algorithm could be adapted to run more efficiently for binary causal worlds.

P_E	Nodes in original network	Avg nodes in W-projection
.1	10	2.27
	20	3.32
	30	5.29
.3	10	5.14
	20	13.16
	30	22.76

Table 1: W-Projection Size (avg of 2000 nets)

N	Brute Force		BF w/ Pruning		IP	
	Avg nodes gen.	Avg Δt (sec)	Avg nodes gen.	Avg Δt (sec)	Avg nodes gen.	Avg Δt (sec)
2	1	<1	1	<1	1	<1
3	4	<1	4	<1	3	<1
4	22	<1	15	<1	7	<1
5	91	<1	53	<1	18	<1
6	304	<1	127	<1	24	<1
7	1416	<1	484	<1	51	<1
8	5413	1	1253	<1	106	<1
9	19692	4	3963	<1	199	<1
10	81490	18	15283	3	406	<1
11	-	-	52142	14	851	<1
12	-	-	-	-	1767	1
13	-	-	-	-	3453	2
14	-	-	-	-	7539	5
15	-	-	-	-	19119	16
16	-	-	-	-	40924	38
17	-	-	-	-	76207	81
18	-	-	-	-	248152	272

Table 2: Algorithm Performance Comparison

Acknowledgments

The author would like to thank Judea Pearl, Richard Korf, and the anonymous reviewers for their helpful advice.

References

Eiter, T., and Lukasiewicz, T. 2001. Complexity results for structure-based causality. *In Proceedings of the International Joint Conference on Artificial Intelligence* 35–40.

Halpern, J., and Pearl, J. 2001. Causes and explanations: A structural-model approach – part i: Causes. *In Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence* 411–420.

Hopkins, M. 2002a. A proof of the conjunctive cause conjecture in 'causes and explanations: A structural-model approach'. Technical Report R–306, UCLA Cognitive Systems Laboratory.

Hopkins, M. 2002b. Strategies for determining causes of events. Technical Report R–302–L, UCLA Cognitive Systems Laboratory.

Nebel, B. 1996. Artificial intelligence: A computational perspective. *Principles of Knowledge Representation* 237–266.

Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge University Press.