Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

The Computational Complexity of Structure-Based Causality

Gadi Aleksandrowicz
IBM Research Lab,
Haifa, Israel
gadia@il.ibm.com

Hana Chockler
Department of Informatics,
King's College,
London, UK
hana.chockler@kcl.ac.uk

Joseph Y. Halpern
Computer Science Department,
Cornell University,
Ithaca, NY, U.S.A.
halpern@cs.cornell.edu

Alexander Ivrii
IBM Research Lab,
Haifa, Israel
alexi@il.ibm.com

Abstract

Halpern and Pearl introduced a definition of actual causality; Eiter and Lukasiewicz showed that computing whether $X = x$ is a cause of $Y = y$ is *NP*-complete in binary models (where all variables can take on only two values) and Σ_2^P-complete in general models. In the final version of their paper, Halpern and Pearl slightly modified the definition of actual cause, in order to deal with problems pointed by Hopkins and Pearl. As we show, this modification has a nontrivial impact on the complexity of computing actual cause. To characterize the complexity, a new family D_k^P, $k = 1, 2, 3, \ldots$, of complexity classes is introduced, which generalizes the class D^P introduced by Papadimitriou and Yannakakis (D^P is just D_1^P). We show that the complexity of computing causality under the updated definition is D_2^P-complete.

Chockler and Halpern extended the definition of causality by introducing notions of *responsibility* and *blame*. The complexity of determining the degree of responsibility and blame using the original definition of causality was completely characterized. Again, we show that changing the definition of causality affects the complexity, and completely characterize it using the updated definition.

1 Introduction

There have been many attempts to define *causality* going back to Hume (1739), and continuing to the present (see, for example, (Collins, Hall, and Paul 2004; Pearl 2000) for some recent work). The standard definitions of causality are based on counterfactual reasoning. In this paper, we focus on one such definition, due to Halpern and Pearl, that has proved quite influential recently.

The definition was originally introduced in 2001 (Halpern and Pearl 2001), but then modified in the final journal version (Halpern and Pearl 2005) to deal with problems pointed out by Hopkins and Pearl (2003). (For ease of reference, we call these definitions "the original HP definition" and "the updated HP definition" in the sequel.) In general, what can be a cause in both the original HP definition and the updated definition is a conjunction of the form $X_1 \leftarrow x_1 \wedge \ldots \wedge X_k \leftarrow x_k$, abbreviated $\vec{X} \leftarrow \vec{x}$; what is caused can be an arbitrary Boolean combination φ of formulas of the form $Y = y$. This should be thought of as saying that setting X_1 to x_1 and ... and setting X_k to x_k results in φ being true. As shown by Eiter and Lukasiewicz (2002) and Hopkins (2001), under the original HP definition, we can always take causes to be single conjuncts. However, as shown by Halpern (2008), this is not the case for the updated HP definition.

Using the fact that causes can be taken to be single conjuncts, Eiter and Lukasiewicz(2002) showed that deciding causality (that is, deciding whether $X = x$ is a cause of φ) is *NP*-complete in binary models (where all variables can take on only two values) and Σ_2^P-complete in general models. As we show here, this is no longer the case for the updated HP definition. Indeed, we completely characterize the complexity of causality for the updated HP definition. To do so, we introduce a new family of complexity classes that may be of independent interest. Papadimitriou and Yannakakis (1984) introduced the complexity class D^P, which consists of all languages L_3 such that there exists a language L_1 in *NP* and a language L_2 in co-*NP* such that $L_3 = L_1 \cap L_2$. We generalize this by defining D_k^P to consist of all languages L_3 such that there exists a language $L_1 \in \Sigma_k^P$ and a language $L_2 \in \Pi_k^P$ such that $L_3 = L_1 \cap L_2$.

Since Σ_1^P is *NP* and Π_1^P is co-*NP*, D_1^P is Papadimitriou and Yannakakis's D^P. We then show that deciding causality under the updated HP definition is D_2^P complete. Papadimitriou and Yannakakis (1984) showed that a number of problems of interest were D^P complete, both for binary and general causal models. To the best of our knowledge, this is the first time that a natural problem has been shown to be complete for D_2^P.

Although, in general, causes may not be single conjuncts, as observed by Halpern (2008), in many cases (in particular, in all the standard examples studied in the literature), they are. In an effort to understand the extent to which the difficulty in deciding causality stems from the fact that causes may require several conjuncts, we consider what we call the

singleton cause problem; that is, the problem of deciding if $X = x$ is a cause of φ (i.e., where there is only a single conjunct in the cause). We show that the singleton cause problem is simpler than the general causality problem (unless the polynomial hierarchy collapses): it is Σ_2^P complete for both binary and general causal models. Thus, if we restrict to singleton causes, the complexity of deciding causality in general models is the same under the original and the updated HP definition, but in binary models, it is still simpler under the original HP definition.

Causality is a "0–1" concept; $\vec{X} = \vec{x}$ is either a cause of φ or it is not. Now consider two voting scenarios: in the first, Mr. G beats Mr. B by a vote of 11–0. In the second, Mr. G beats Mr. B by a vote of 6–5. According to both the original and the updated HP definition, all the people who voted for Mr. G are causes of him winning. While this does not seem so unreasonable, it does not capture the intuition that each voter for Mr. G is more critical to the victory in the case of the 6–5 vote than in the case of the 11–0 vote. The notion of *degree of responsibility*, introduced by Chockler and Halpern (2004), does so. The idea is that the degree of responsibility of $X = x$ for φ is $1/(k+1)$, where k is the least number of changes that have to be made in order to make $X = x$ critical. In the case of the 6–5 vote, no changes have to be made to make each voter for Mr. G critical for Mr. G's victory; if he had not voted for Mr. G, Mr. G would not have won. Thus, each voter has degree of responsibility 1 (i.e., $k = 0$). On the other hand, in the case of the 11–0 vote, for a particular voter to be critical, five other voters have to switch their votes; thus, $k = 5$, and each voter's degree of responsibility is $1/6$. This notion of degree of responsibility has been shown to capture (at a qualitative level) the way people allocate responsibility (Gerstenberg and Lagnado 2010; Lagnado, Gerstenberg, and Zultan 2013).

Chockler and Halpern further extended the notion of degree of responsibility to *degree of blame*. Formally, the degree of blame is the expected degree of responsibility. This is perhaps best understood by considering a firing squad with ten excellent marksmen. Only one of them has live bullets in his rifle; the rest have blanks. The marksmen do not know which of them has the live bullets. The marksmen shoot at the prisoner and he dies. The only marksman that is the cause of the prisoner's death is the one with the live bullets. That marksman has degree of responsibility 1 for the death; all the rest have degree of responsibility 0. However, each of the marksmen has degree of blame $1/10$.The complexity of determining the degree of responsibility and blame using the original definition of causality was completely characterized (Chockler and Halpern 2004; Chockler, Halpern, and Kupferman 2008). Again, we show that changing the definition of causality affects the complexity, and completely characterize the complexity of determining the degree of responsibility and blame with the updated definition.

The rest of this paper is organized as follows. In Section 2, we review the relevant definitions of causality. In Section 3, we briefly review the relevant definitions from complexity theory and define the complexity classes D_k^P. In Section 4 we prove our results on complexity of causality.[1]

2 Causal Models and Causality: A Review

In this section, we review the details of Halpern and Pearl's definition of causal models and causality, describing both the original definition and the updated definition. This material is largely taken from (Halpern and Pearl 2005), to which we refer the reader for further details.

2.1 Causal models

A *signature* is a tuple $\mathcal{S} = \langle \mathcal{U}, \mathcal{V}, \mathcal{R} \rangle$, where $\mathcal{U}$ is a finite set of *exogenous* variables, $\mathcal{V}$ is a finite set of *endogenous* variables, and $\mathcal{R}$ associates with every variable $Y \in \mathcal{U} \cup \mathcal{V}$ a finite nonempty set $\mathcal{R}(Y)$ of possible values for Y. Intuitively, the exogenous variables are ones whose values are determined by factors outside the model, while the endogenous variables are ones whose values are ultimately determined by the exogenous variables. A *causal model* over signature $\mathcal{S}$ is a tuple $M = \langle \mathcal{S}, \mathcal{F} \rangle$, where $\mathcal{F}$ associates with every endogenous variable $X \in \mathcal{V}$ a function F_X such that $F_X : (\times_{U \in \mathcal{U}} \mathcal{R}(U) \times (\times_{Y \in \mathcal{V} \setminus \{X\}} \mathcal{R}(Y))) \rightarrow \mathcal{R}(X)$. That is, F_X describes how the value of the endogenous variable X is determined by the values of all other variables in $\mathcal{U} \cup \mathcal{V}$. If $\mathcal{R}(Y)$ contains only two values for each $Y \in \mathcal{U} \cup \mathcal{V}$, then we say that M is a *binary causal model*.

We can describe (some salient features of) a causal model M using a *causal network*. A causal network is a graph with nodes corresponding to the random variables in $\mathcal{V}$ and an edge from a node labeled X to one labeled Y if F_Y depends on the value of X. Intuitively, variables can have a causal effect only on their descendants in the causal network; if Y is not a descendant of X, then a change in the value of X has no affect on the value of Y. For ease of exposition, we restrict attention to what are called *recursive* models. These are ones whose associated causal network is a directed acyclic graph (that is, a graph that has no cycle of edges). Actually, it suffices for our purposes that, for each setting $\vec{u}$ for the variables in $\mathcal{U}$, there is no cycle among the edges of the causal network. We call a setting $\vec{u}$ for the variables in $\mathcal{U}$ a *context*. It should be clear that if M is a recursive causal model, then there is always a unique solution to the equations in M, given a context.

The equations determined by $\{F_X : X \in \mathcal{V}\}$ can be thought of as representing processes (or mechanisms) by which values are assigned to variables. For example, if $F_X(Y, Z, U) = Y + U$ (which we usually write as $X = Y + U$), then if $Y = 3$ and $U = 2$, then $X = 5$, regardless of how Z is set. This equation also gives counterfactual information. It says that, in the context $U = 4$, if Y were 4, then X would be 8, regardless of what value X and Z actually take in the real world. That is, if $U = 4$ and the value of Y were forced to be 4 (regardless of its actual value), then the value of X would be 8.

While the equations for a given problem are typically obvious, the choice of variables may not be. Consider the following example (due to Hall (2004)), showing that the

[1] Missing proof details can be found at http://www.cs.cornell.edu/home/halpern/papers/newcause.pdf.

choice of variables influences the causal analysis. Suppose that Suzy and Billy both pick up rocks and throw them at a bottle. Suzy's rock gets there first, shattering the bottle. Since both throws are perfectly accurate, Billy's would have shattered the bottle had Suzy not thrown.

In this case, a naive model might have an exogenous variable U that encapsulates whatever background factors cause Suzy and Billy to decide to throw the rock (the details of U do not matter, since we are interested only in the context where U's value is such that both Suzy and Billy throw), a variable *ST* for Suzy throws ($ST = 1$ if Suzy throws, and $ST = 0$ if she doesn't), a variable *BT* for Billy throws, and a variable *BS* for bottle shatters. In the naive model, whose graph is given in Figure 1, BS is 1 if one of *ST* and *BT* is 1.

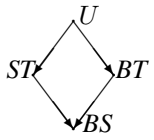

Figure 1: A naive model for the rock-throwing example.

This causal model does not distinguish between Suzy and Billy's rocks hitting the bottle simultaneously and Suzy's rock hitting first. A more sophisticated model might also include variables *SH* and *BH*, for Suzy's rock hits the bottle and Billy's rock hits the bottle. Clearly *BS* is 1 iff one of *SH* and *BH* is 1. However, now, *SH* is 1 if *ST* is 1, and $BH = 1$ if $BT = 1$ and $SH = 0$. Thus, Billy's throw hits if Billy throws *and* Suzy's rock doesn't hit. This model is described by the following graph, where we implicitly assume a context where Suzy throws first, so there is an edge from *SH* to *BH*, but not one in the other direction (and omit the exogenous variable).

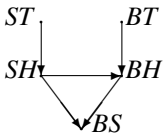

Figure 2: A better model for the rock-throwing example.

Given a causal model $M = (\mathcal{S}, \mathcal{F})$, a (possibly empty) vector $\vec{X}$ of variables in $\mathcal{V}$, and a vector $\vec{x}$ of values for the variables in $\vec{X}$, we define a new causal model, denoted $M_{\vec{X} \leftarrow \vec{x}}$, which is identical to M, except that the equation for the variables $\vec{X}$ in $\mathcal{F}$ is replaced by $\vec{X} = \vec{x}$. Intuitively, this is the causal model that results when the variables in $\vec{X}$ are set to $\vec{x}$ by some external action that affects only the variables in $\vec{X}$ (and overrides the effects of the causal equations). For example, if M is the more sophisticated model for the rock-throwing example, then $M_{ST \leftarrow 0}$ is the model where Suzy doesn't throw.

Given a signature $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$, a formula of the form $X = x$, for $X \in \mathcal{V}$ and $x \in \mathcal{R}(X)$, is called a *primitive event*. A *basic causal formula* has the form $[Y_1 \leftarrow y_1, \ldots, Y_k \leftarrow y_k]\varphi$, where φ is a Boolean combination of primitive events; $Y_1, \ldots, Y_k$ are distinct variables in $\mathcal{V}$; and $y_i \in \mathcal{R}(Y_i)$. Such a formula is abbreviated as $[\vec{Y} \leftarrow \vec{y}]\varphi$. The special case where $k = 0$ is abbreviated as φ. Intuitively, $[Y_1 \leftarrow y_1, \ldots, Y_k \leftarrow y_k]\varphi$ says that φ holds in the counterfactual world that would arise if Y_i is set to y_i, for $i = 1, \ldots, k$. A *causal formula* is a Boolean combination of basic causal formulas.

A causal formula φ is true or false in a causal model, given a *context*. We write $(M, \vec{u}) \models \varphi$ if φ is true in causal model M given context $\vec{u}$. $(M, \vec{u}) \models [\vec{Y} \leftarrow \vec{y}](X = x)$ if the variable X has value x in the unique (since we are dealing with recursive models) solution to the equations in $M_{\vec{Y} \leftarrow \vec{y}}$ in context $\vec{u}$ (i.e., the unique vector of values for the exogenous variables that simultaneously satisfies all equations $F_Z^{\vec{Y} \leftarrow \vec{y}}$, $Z \in \mathcal{V} - \vec{Y}$, with the variables in $\mathcal{U}$ set to $\vec{u}$). We extend the definition to arbitrary causal formulas in the obvious way.

2.2 Causality

We now review the updated HP definition of causality.

Definition 2.1 $\vec{X} = \vec{x}$ *is a* cause *of* φ *in* $(M, \vec{u})$ *if the following three conditions hold:*

AC1. $(M, \vec{u}) \models (\vec{X} = \vec{x}) \wedge \varphi$.

AC2. *There exist a partition* $(\vec{Z}, \vec{W})$ *of* $\mathcal{V}$ *with* $\vec{X} \subseteq \vec{Z}$ *and some setting* $(\vec{x}', \vec{w})$ *of the variables in* $(\vec{X}, \vec{W})$ *such that if* $(M, \vec{u}) \models Z = z^*$ *for* $Z \in \vec{Z}$*, then*

(a) $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}]\neg\varphi$.

(b) $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W}' \leftarrow \vec{w}, \vec{Z}' \leftarrow \vec{z}^*]\varphi$ *for all subsets* $\vec{Z}'$ *of* $\vec{Z} \setminus \vec{X}$ *and all subsets* $\vec{W}'$ *of* $\vec{W}$*, where we abuse notation and write* $\vec{W}' \leftarrow \vec{w}$ *to denote the assignment where the variables in* $\vec{W}'$ *get the same values as they would in the assignment* $\vec{W} \leftarrow \vec{w}$*, and similarly for* $\vec{Z}' \leftarrow \vec{z}^*$*. That is, setting any subset* $\vec{W}'$ *of* $\vec{W}$ *to the values in* $\vec{w}$ *should have no effect on* φ *as long as* $\vec{X}$ *has the value* $\vec{x}$*, even if all the variables in an arbitrary subset of* $\vec{Z}$ *are set to their original values in the context* $\vec{u}$*.*

AC3. $(\vec{X} = \vec{x})$ *is minimal; no subset of* $\vec{X}$ *satisfies AC2.*

If $\vec{X}$ is a singleton, then $X = x$ is said to be a *singleton cause* of φ in $(M, \vec{u})$.

AC1 just says that A cannot be a cause of B unless both A and B are true. The core of this definition lies in AC2. Informally, the variables in $\vec{Z}$ should be thought of as describing the "active causal process" from X to φ. These are the variables that mediate between X and φ. AC2(a) is reminiscent of the traditional counterfactual criterion, according to which $X = x$ is a cause of φ if changing the value of X results in φ being false. However, AC2(a) is more permissive than the traditional criterion; it allows the dependence of φ on X to be tested under special *structural contingencies*, in which the variables $\vec{W}$ are held constant at some setting $\vec{w}$. AC2(b) is an attempt to counteract the "permissiveness" of AC2(a) with regard to structural contingencies. Essentially, it ensures that X alone suffices to bring about the change

from φ to $\neg\varphi$; setting $\vec{W}$ to $\vec{w}$ merely eliminates spurious side effects that tend to mask the action of X.

To understand the role of AC2(b), consider the rock-throwing example again. Let M be the model in Figure 1, and let $\vec{u}$ be the context where both Suzy and Billy throw. It is easy to see that both Suzy and Billy are causes of the bottle shattering in $(M, \vec{u})$: Let $\vec{Z} = \{ST, BS\}$, and consider the structural contingency where Billy doesn't throw ($BT = 0$). Clearly $(M, U) \models [ST \leftarrow 0, BT \leftarrow 0](BS = 0)$ and $(M, u) \models [ST \leftarrow 1, BT \leftarrow 0](BS = 1)$, so Suzy is a cause of the bottle shattering. A symmetric argument shows that Billy is also a cause.

But now consider the model M' described in Figure 2; again, u is the context where both Suzy and Billy throw. It is still the case that Suzy is a cause of the bottle shattering in (M', u). We can take $\vec{W} = \{BT\}$ and again consider the contingency where Billy doesn't throw. However, Billy is *not* a cause of the bottle shattering in (M', u). For suppose that we now take $\vec{W} = \{ST\}$ and consider the contingency where Suzy doesn't throw. Clearly AC2(a) holds, since if Billy doesn't throw (under this contingency), then the bottle doesn't shatter. However, AC2(b) does not hold. Since $BH \in \vec{Z}$, if we set BH to 0 (its original value), then AC2(b) would require that $(M', u) \models [BT \leftarrow 1, ST \leftarrow 0, BH \leftarrow 0](BS = 1)$, but this is not the case. Similar arguments show that no other choice of $(\vec{Z}, \vec{W})$ makes Billy's throw a cause of the bottle shattering in (M', u).

The original HP definition differs from the updated definition in only one respect. Rather than requiring that $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W}' \leftarrow \vec{w}, \vec{Z}' \leftarrow \vec{z}^*]\varphi$ for all subsets $\vec{W}'$ of $\vec{W}$, it was required to hold only for $\vec{W}$. That is, the following condition was used instead of AC2(b).

AC2(b′) $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W} \leftarrow \vec{w}, \vec{Z}' \leftarrow \vec{z}^*]\varphi$ for all subsets $\vec{Z}'$ of $\vec{Z}$.

The requirement for AC2(b) to hold for all subsets of W in the updated definition prevents situations where W "conceals other causes for φ". The role of this requirement is perhaps best understood by considering the following example, due to Hopkins and Pearl (2003) (the description is taken from (Halpern and Pearl 2005)): Suppose that a prisoner dies either if A loads B's gun and B shoots, or if C loads and shoots his gun. Taking D to represent the prisoner's death and making the obvious assumptions about the meaning of the variables, we have that $D = (A \wedge B) \vee C$. Suppose that in the actual context u, A loads B's gun, B does not shoot, but C does load and shoot his gun, so that the prisoner dies. That is, $A = 1$, $B = 0$, and $C = 1$. Clearly $C = 1$ is a cause of $D = 1$. We would not want to say that $A = 1$ is a cause of $D = 1$, given that B did not shoot (i.e., given that $B = 0$). However, with AC2(b′), $A = 1$ is a cause of $D = 1$. For we can take $\vec{W} = \{B, C\}$ and consider the contingency where $B = 1$ and $C = 0$. It is easy to check that AC2(a) and AC2(b′) hold for this contingency, so under the original HP definition, $A = 1$ is a cause of $D = 1$. However, AC2(b) fails in this case, since $(M, u) \models [A \leftarrow 1, C \leftarrow 0](D = 0)$. The key point is that AC2(b) says that for $A = 1$ to be a cause of $D = 1$, it must be the case that $D = 0$ if only some of the values in $\vec{W}$ are set to $\vec{w}$. That means that the other variables get the same value as they do in the actual context; in this case, by setting only A to 1 and leaving B unset, B takes on its original value of 0, in which case $D = 0$. AC2(b′) does not consider this case.

Using AC2(b) rather than AC2(b′) has been shown to have a significant benefit (and to lead to more intuitive results) when causality is applied to program verification, with the goal of understanding what in the code is the cause of a program not satisfying its specification (Beer et al. 2012).

3 Relevant Complexity Classes

In this section, we briefly recall the definitions of the complexity classes that we need for our results, and define the complexity class D_2^k.

Recall that the *polynomial hierarchy* is a hierarchy of complexity classes that generalize the classes *NP* and co-*NP*. Let $\Sigma_1^P = NP$ and $\Pi_1^P =$ co-*NP*. For $i > 1$, define $\Sigma_i^P = NP^{\Sigma_{i-1}^P}$ and $\Pi_i^P = (\text{co-}NP)^{\Sigma_{i-1}^P}$, where, in general, X^Y denotes the class of problems solvable by a Turing machine in class A augmented with an oracle for a problem complete for class B. (See (Meyer and Stockmeyer 1972; Stockmeyer 1977) for more details and intuition.)

We now define the classes D_k^P as follows.

Definition 3.1 *For $k = 1, 2, \ldots$,*

$$D_k^P = \{L : \exists L_1, L_2 : L_1 \in \Sigma_k^P, L_2 \in \Pi_k^P, L = L_1 \cap L_2\}.$$

For $k = 1$, the class D_1^P is the well-known complexity class D^P, defined by Papadimitriou and Yannakakis (1984). It contains *exact* problems such as the language of pairs $\langle G, k\rangle$, where G is a graph that has a maximal clique of size exactly k. As usual, we say that a language L is D_k^P complete if it is in D_k^P and is the "hardest" language in D_k^P, in the sense that there is a polynomial time reduction from any language $L' \in D_k^P$ to L.

Recall that a *q*uantified Boolean formula (QBF) is a generalization of a propositional formula, where some propositional variables are quantified. Thus, for example, $\exists x \forall y(x \vee y)$ is a QBF. A *closed* QBF (CQBF) is one where there are no free propositional variables. A CQBF is either true or false, independent of the truth assignment. The "canonical" languages complete for Σ_2^k and Π_2^k consist of the CQBFs with k alternations of quantifiers starting with $\exists$ (resp., $\forall$) that are true. In particular, let

$$\Sigma_2^P(\text{SAT}) = \{\exists \vec{X} \forall \vec{Y} \varphi \mid \exists \vec{X} \forall \vec{Y} \varphi \text{ is a CQBF, } \exists \vec{X} \forall \vec{Y} \varphi = \textbf{true}\}$$
$$\Pi_2^P(\text{SAT}) = \{\forall \vec{X} \exists \vec{Y} \varphi \mid \forall \vec{X} \exists \vec{Y} \varphi \text{ is a CQBF, } \forall \vec{X} \exists \vec{Y} \varphi = \textbf{true}\}.$$

$\Sigma_2^P(\text{SAT})$ is complete for Σ_2^P and $\Pi_2^P(\text{SAT})$ is complete for Π_2^P (Wrathall 1976).

The following lemma provides a useful condition sufficient for a language to be D_k^P-complete.

Lemma 3.2 *If L_1 is Σ_k^P-complete and L_2 is Π_k^P-complete, then $L_3 = L_1 \cap L_2$ is D_k^P-complete.*

Proof: The fact that L_3 is in D_k^P is immediate from the definition of D_k^P. For hardness, let L_3' be a language in D_k^P. Then there exist L_1' and L_2' such that $L_1' \in \Sigma_k^P, L_2' \in \Pi_k^P$, and $L' = L_1' \cap L_2'$. Let f be a polynomial-time reduction from L_1' to L_1, and let g be a polynomial-time reduction from L_2' to L_2 (the existence of such reductions f and g follows from the fact that L_1 and L_2 are Σ_k^P-complete and Π_k^P-complete, respectively). Then, $\langle f, g\rangle$ is a polynomial-time reduction from L_3' to L_3, as required. □

Essentially the same argument shows that if L_1 is Σ_k^P-hard and L_2 is Π_k^P-hard, then $L_3 = L_1 \cap L_2$ is D_k^P-hard.

Determining whether $\vec{X} = \vec{x}$ is a cause of φ in (M, u) is a decision problem: we define a language and try to determine whether a particular tuple is in that language. (See Section 4 for the formal definition.) Determining degree of responsibility and blame is a different type of problem, since we are determining which number represents the degree of responsibility (resp., blame). Formally, these are *function problems*. For ease of exposition, we restrict attention to functions from some strings over some fixed language Σ to strings over Σ (i.e., we are considering functions from Σ^* to Σ^*). For a complexity class A in the polynomial hierarchy, $\text{FP}^{\text{A}[\log n]}$ consists of all functions that can be computed by a polynomial-time Turing machine with an A-oracle which on input x asks a total of $O(\log |x|)$ queries (Papadimitriou 1984). A function $f(x)$ is $\text{FP}^{\text{A}[\log n]}$hard iff for every function $g(x)$ in $\text{FP}^{\text{A}[\log n]}$ there exist polynomially computable functions $R, S : \Sigma^* \to \Sigma^*$ such that $g(x) = S(f(R(x)))$. A function $f(x)$ is complete in $\text{FP}^{\text{A}[\log n]}$ iff it is in $\text{FP}^{\text{A}[\log n]}$ and is $\text{FP}^{\text{A}[\log n]}$-hard.

Finally, for a complexity class A in polynomial hierarchy, $\text{FP}_{||}^{\text{A}}$is the class of functions that can be computed by a polynomial-time Turing machine with parallel (i.e., non-adaptive) queries to an A-oracle. (For background on these complexity classes, see (Jenner and Toran 1995; Johnson 1990).)

4 Complexity for the Updated HP Definition

In this section, we prove our results on the complexity of deciding causality. We start by defining the problem formally. In the definitions, M stands for a causal model, $\vec{u}$ is a context, $\vec{X}$ is a subset of variables of M, and $\vec{x}$ is the set of values of $\vec{X}$ in $(M, \vec{u})$:

$$L_{\text{cause}} = \{\langle M, \vec{u}, \varphi, \vec{X}, \vec{x}\rangle : (\vec{X} = \vec{x}) \text{ is a cause of } \varphi \text{ in } (M, \vec{u})\}.$$

One of our goals is to understand the cause of the complexity of computing causality. Towards this end, it is useful to define two related languages:

$$L_{\text{AC2}} = \{\langle M, \vec{u}, \varphi, \vec{X}, \vec{x}\rangle : (\vec{X} = \vec{x}) \text{ satisfies conditions AC1 and AC2 of Def. 2.1 for } \varphi \text{ in } (M, \vec{u})\},$$
$$L_{\text{AC3}} = \{\langle M, \vec{u}, \varphi, \vec{X}, \vec{x}\rangle : (\vec{X} = \vec{x}) \text{ satisfies conditions AC1 and AC3 of Def. 2.1 for } \varphi \text{ in } (M, \vec{u})\}.$$

It is easy to see that $L_{\text{cause}} = L_{\text{AC2}} \cap L_{\text{AC3}}$.

Let L_{cause}^1 be the subset of L_{cause} where $\vec{X}$ and $\vec{x}$ are singletons; this is the singleton causality problem. We can similarly define L_{AC2}^1 and L_{AC3}^1. Again, we have $L_{\text{cause}}^1 = L_{\text{AC2}}^1 \cap L_{\text{AC3}}^1$, but, in fact, we have $L_{\text{cause}}^1 = L_{\text{AC2}}^1$, since $L_{\text{AC2}}^1 \subseteq L_{\text{AC3}}^1$; for singleton causality, the minimality condition AC3 trivially holds.

We denote by L_{cause}^B the language of causality for binary causal models (i.e., where the models M in the tuple are binary models), and by L_{AC2}^B and L_{AC3}^B the languages L_{AC2} and L_{AC3} restricted to binary causal models. Again we have that $L_{\text{cause}}^B = L_{\text{AC2}}^B \cap L_{\text{AC3}}^B$. And again, we can define $L_{\text{cause}}^{B,1}$, $L_{\text{AC2}}^{B,1}$, and $L_{\text{AC3}}^{B,1}$, and we have $L_{\text{cause}}^{B,1} = L_{\text{AC2}}^{B,1}$.

We start by considering singleton causality. As we observed, Eiter and Lukasiewicz (2002) and Hopkins (2001) showed that, with the original HP definition, singleton causality and causality coincide. However, for the updated definition, Halpern (2008) showed that it is in fact possible to have minimal causes that are not singletons. Thus, we consider singleton causality and general causality separately. We can clarify where the complexity lies by considering L_{AC2} (and its sublanguages) and L_{AC3} (and its sublanguages) separately.

Theorem 4.1 *The languages L_{AC2}, L_{AC2}^1, $L_{AC2}^{B,1}$, and L_{AC2}^1 are Σ_2^P-complete.*

Proof outline: To show all these languages are in Σ^P, given a tuple $\langle M, \vec{u}, \varphi, \vec{X}, \vec{x}\rangle$, checking that AC1 holds, that is, checking that $(M, \vec{u}) \models \vec{X} = \vec{x} \wedge \varphi$, can be done in time polynomial in the size of M, $|\vec{X}|$, and $|\varphi|$ (the length of φ as a string of symbols). For AC2, we need only guess the set $\vec{W}$ and the assignment $\vec{w}$. The check that assigning $\vec{w}$ to $\vec{W}$ and x' to X indeed falsifies φ is polynomial, and we use an *NP* oracle to check that for all subsets of $\vec{W}$ and all subsets of $\vec{Z}$, condition AC2(b) holds. (The argument is quite similar to Eiter and Lukasiewicz's argument that causality is in Σ_2^P for general models with the original HP definition.)

For hardness, it clearly suffices to show that $L_{\text{AC2}}^{B,1}$ is Σ_2^P-hard. We do this by reducing $\Sigma_2^P(\text{SAT})$ to $L_{\text{AC2}}^{B,1}$. Given a CQBF formula $\exists\vec{X}\forall\vec{Y}\varphi$, we show that we can efficiently construct a causal formula ψ, model M, and context u such that $\exists\vec{X}\forall\vec{Y}\varphi =$ **true** iff $(M, u, \psi, A, 0) \in L_{\text{AC2}}^{B,1}$. We leave details to the full paper. □

Since, as we have observed, AC3 is vacuous in the case of singleton causality, it follows that singleton causality is Σ_2^P-complete.

Corollary 4.2 *L_{cause}^1 and $L_{cause}^{B,1}$ are Σ_2-complete.*

We now show that things are harder if we do not restrict to binary causal models (unless the polynomial hierarchy collapses). As a first step, we consider the complexity of L_{AC3} and L_{AC3}^B.

Theorem 4.3 *L_{AC3} and L_{AC3}^B are Π_2^P-complete.*

Proof outline: The fact that L_{AC3} and L^B_{AC3} are in Π^P_2 is straightforward. Again, given a tuple $\langle M, \vec{u}, \varphi, \vec{X}, \vec{x}\rangle$, we can check that AC1 holds in polynomial time. For AC3, we need to check that for all strict subsets $\vec{X}'$ of $\vec{X}$, AC2 fails. Since checking AC2 is in Σ^P_2, checking that it fails is in Π^P_2. Checking that it fails for all strict subsets $\vec{X}'$ keeps it in Π^P_2 (since it just adds one more universal quantifier).

To prove that these languages are Π^P_2-hard, we show that we can reduce Π^P_2(SAT) to L^B_{AC3}. The proof is similar in spirit to the proof of Theorem 4.1; we leave details to the full paper. □

We are now ready to prove our main result.

Theorem 4.4 *L_{cause} and L^B_{cause} are D^P_2-complete.*

Proof: Membership of L_{cause} (and hence also L^B_{cause}) in D^P_2 follows from the fact that $L_{\text{cause}} = L_{\text{AC2}} \cap L_{\text{AC3}}$, $L_{\text{AC2}} \in \Sigma^P_2$, and $L_{\text{AC3}} \in \Pi^P_2$. The fact that L^B_{cause} (and hence also L_{cause}) are D^P_2-hard follows from Lemma 3.2 and Theorems 4.1 and 4.3. □

5 Responsibility and Blame

In this section, we review the definitions of responsibility and blame and characterize their complexity. See Chockler and Halpern (2004) for more intuition and details.

5.1 Responsibility

The definition of responsibility given by Chockler and Halpern (2004) was given based on the original HP definition of causality, and thus assumed that causes were always single conjuncts. It is straightforward to extend it to allow causes to have arbitrarily many conjuncts.

Definition 5.1 *The* degree of responsibility of $\vec{X} = \vec{x}$ for φ in $(M, \vec{u})$, *denoted* $\text{dr}((M, \vec{u}), (\vec{X} = \vec{x}), \varphi)$, *is* 0 *if* $\vec{X} = \vec{x}$ *is not a cause of* φ *in* $(M, \vec{u})$*; it is* $1/(k + 1)$ *if* $\vec{X} = \vec{x}$ *is a cause of* φ *in* $(M, \vec{u})$ *and there exists a partition* $(\vec{Z}, \vec{W})$ *and setting* $(\vec{x}', \vec{w})$ *for which AC2 holds such that (a)* k *variables in* $\vec{W}$ *have different values in* $\vec{w}$ *than they do in the context* $\vec{u}$ *and (b) there is no partition* $(\vec{Z}', \vec{W}')$ *and setting* $(\vec{x}'', \vec{w}')$ *satisfying AC2 such that only* $k' < k$ *variables have different values in* $\vec{w}'$ *than they do the context* $\vec{u}$*.*

Intuitively, $dr((M, \vec{u}), (\vec{X} = \vec{x}), \varphi)$ measures the minimal number of changes that have to be made in $\vec{u}$ in order to make φ counterfactually depend on $\vec{X}$, provided the conditions on the subsets of $\vec{W}$ and $\vec{Z}$ are satisfied (see also the voting example from the introduction). If there is no partition of $\mathcal{V}$ to $(\vec{Z}, \vec{W})$ that satisfies AC2, or $(\vec{X} = \vec{x})$ does not satisfy AC3 for φ in $(M, \vec{u})$, then the minimal number of changes in $\vec{u}$ in Definition 5.1 is taken to have cardinality ∞, and thus the degree of responsibility of $(\vec{X} = \vec{x})$ is 0 (and hence it is not a cause).

In the original HP model, it was shown that computing responsibility is $\text{FP}^{\text{NP}[\log n]}$-complete in binary causal models (Chockler, Halpern, and Kupferman 2008) and $\text{FP}^{\Sigma^P_2[\log n]}$-complete in general causal models (Chockler and Halpern 2004). We now characterize the complexity of computing responsibility in the updated HP definition.

Theorem 5.2 *Computing the degree of responsibility is $FP^{\Sigma^P_2[\log n]}$-complete for singleton causes in binary and general causal models.*

Proof outline: The proof is quite similar to the proof in (Chockler and Halpern 2004). We prove membership by describing an algorithm in $\text{FP}^{\Sigma^P_2[\log n]}$ for computing the degree of responsibility. Roughly speaking, the algorithm queries an oracle for the language $\mathcal{R} = \{(\langle (M, \vec{u}), (X = x), \varphi, i\rangle$ such that $\langle (M, \vec{u}), (X = x), \varphi\rangle \in L_{cause}$ and the degree of responsibility of $(X = x)$ for φ is at least $i\}$. It is easy to see that $\mathcal{R}$ is in Σ^P_2 by using Corollary 4.2. The algorithm for computing the degree of responsibility performs a binary search on the value of $dr((M, \vec{u}), (X = x), \varphi)$, each time dividing the range of possible values for the degree of responsibility by 2 according to the answer of $\mathcal{R}$. The number of possible candidates for the degree of responsibility is bounded by the size of the input n, and thus the number of queries is at most $\lceil \log n \rceil$.

For hardness in binary causal models (which implies hardness in general causal models), we provide a reduction from the Σ^P_2-complete problem MINQSAT_2 (Chockler and Halpern 2004) to the degree of responsibility, where $\text{MINQSAT}_2(\exists \vec{X} \forall \vec{Y} \psi)$ is the minimum number of 1's in the satisfying assignment to $\vec{X}$ for $\exists \vec{X} \forall \vec{Y} \psi$ if such an assignment exists, and $|\vec{X}| + 1$ otherwise.

□

Theorem 5.3 *Computing the degree of responsibility is $FP^{D_2[\log n]}$-complete in binary and general causal models.*

5.2 Blame

The definition of blame addresses the situation where there is uncertainty about the true situation or "how the world works". Blame, introduced in (Chockler and Halpern 2004), considers the "true situation" to be determined by the context, and "how the world works" to be determined by the structural equations. An agent's uncertainty is modeled by a pair $(\mathcal{K}, \Pr)$, where $\mathcal{K}$ is a set of pairs of the form $(M, \vec{u})$, where M is a causal model and $\vec{u}$ is a context, and $\Pr$ is a probability distribution over $\mathcal{K}$. A pair $(M, \vec{u})$ is called a *situation*. We think of $\mathcal{K}$ as describing the situations that the agent considers possible before $\vec{X}$ is set to $\vec{x}$. The degree of blame that setting $\vec{X}$ to $\vec{x}$ has for φ is then the expected degree of responsibility of $\vec{X} = \vec{x}$ for φ in $(M_{\vec{X} \leftarrow \vec{x}}, \vec{u})$, taken over the situations $(M, \vec{u}) \in \mathcal{K}$. Note that the situation $(M_{\vec{X} \leftarrow \vec{x}}, \vec{u})$ for $(M, \vec{u}) \in \mathcal{K}$ are those that the agent considers possible after $\vec{X}$ is set to $\vec{x}$.

Definition 5.4 *The* degree of blame of setting $\vec{X}$ to $\vec{x}$ for φ relative to epistemic state $(\mathcal{K}, \Pr)$, *denoted* $\text{db}(\mathcal{K}, \Pr, \vec{X} \leftarrow \vec{x}, \varphi)$, *is*

$$\sum_{(M,\vec{u}) \in \mathcal{K}} \text{dr}((M_{\vec{X} \leftarrow \vec{x}}, \vec{u}), \vec{X} = \vec{x}, \varphi) \Pr((M, \vec{u})).$$

For the original HP definition of cause, Chockler and Halpern (2004) show that computing the degree of blame is complete in $\mathrm{FP}_{||}^{\Sigma_2^P}$ for general and in $\mathrm{FP}_{||}^{\mathrm{NP}}$ for binary causal models. Again, with the updated HP definition, the complexity changes.

Theorem 5.5 *The problem of computing blame in recursive causal models is $FP_{||}^{\Sigma_2^P}$-complete for singleton causes and $FP_{||}^{D_2}$-complete for (general) causes, in binary and general causal models.*

Acknowledgements: Joseph Halpern was supported in part by NSF grants IIS-0911036 and CCF-1214844, AFOSR grant FA9550-08-1-0438, ARO grant W911NF-14-1-0017, and by the DoD Multidisciplinary University Research Initiative (MURI) program administered by AFOSR under grant FA9550-12-1-0040.

References

Beer, I.; Ben-David, S.; Chockler, H.; Orni, A.; and Trefler, R. J. 2012. Explaining counterexamples using causality. *Formal Methods in System Design* 40(1):20–40.

Chockler, H., and Halpern, J. 2004. Responsibility and blame: a structural-model approach. *Journal of Artificial Intelligence Research (JAIR)* 22:93–115.

Chockler, H.; Halpern, J. Y.; and Kupferman, O. 2008. What causes a system to satisfy a specification? *ACM Transactions on Computational Logic* 9(3).

Collins, J.; Hall, N.; and Paul, L. A., eds. 2004. *Causation and Counterfactuals*. Cambridge, Mass.: MIT Press.

Eiter, T., and Lukasiewicz, T. 2002. Complexity results for structure-based causality. *Artificial Intelligence* 142(1):53–89.

Gerstenberg, T., and Lagnado, D. 2010. Spreading the blame: the allocation of responsibility amongst multiple agents. *Cognition* 115:166–171.

Hall, N. 2004. Two concepts of causation. In Collins, J.; Hall, N.; and Paul, L. A., eds., *Causation and Counterfactuals*. Cambridge, Mass.: MIT Press.

Halpern, J. Y., and Pearl, J. 2001. Causes and explanations: A structural-model approach. Part I: Causes. In *Proc. Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI 2001)*, 194–202.

Halpern, J. Y., and Pearl, J. 2005. Causes and explanations: A structural-model approach. Part I: Causes. *British Journal for Philosophy of Science* 56(4):843–887.

Halpern, J. Y. 2008. Defaults and normality in causal structures. In *Principles of Knowledge Representation and Reasoning: Proc. Eleventh International Conference (KR '08)*. 198–208.

Hopkins, M., and Pearl, J. 2003. Clarifying the usage of structural models for commonsense causal reasoning. In *Proc. AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*.

Hopkins, M. 2001. A proof of the conjunctive cause conjecture. Unpublished manuscript.

Hume, D. 1739. *A Treatise of Human Nature*. London: John Noon.

Jenner, B., and Toran, J. 1995. Computing functions with parallel queries to NP. *Theoretical Computer Science* 141:175–193.

Johnson, D. S. 1990. A catalog of complexity classes. In Leeuwen, J. v., ed., *Handbook of Theoretical Computer Science*, volume A. Elsevier Science. chapter 2.

Lagnado, D.; Gerstenberg, T.; and Zultan, R. 2013. Causal responsibility and counterfactuals. *Cognitive Science* 37:1036–1073.

Meyer, A., and Stockmeyer, L. 1972. The equivalence problem for regular expressions with squaring requires exponential time. In *Proc. 13th IEEE Symp. on Switching and Automata Theory*, 125–129.

Papadimitriou, C. H., and Yannakakis, M. 1984. The complexity of facets (and some facets of complexity). *J. Comput. Syst. Sci.* 28(2):244–259.

Papadimitriou, C. H. 1984. The complexity of unique solutions. *Journal of ACM* 31:492–500.

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

Stockmeyer, L. J. 1977. The polynomial-time hierarchy. *Theoretical Computer Science* 3:1–22.

Wrathall, C. 1976. Complete sets and the polynomial-time